Escuela Politécnica Superior

Visualización automática de grandes volúmenes de datos para usuarios inexpertos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Luis Márquez Carpintero

Tutor/es:
Alejandro Mate Morga

Junio 2020

Universitat d'Alacant
Universidad de Alicante

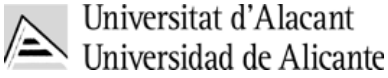
Visualización automática de grandes volúmenes de datos para usuarios inexpertos

TRABAJO FIN DE GRADO

Autor
Luis Márquez Carpintero

Tutor
Alejandro Mate Morga
Departamento de Lenguaje de Sistemas Informáticos

Grado en Ingeniería Informática



5 de junio, 2020

Justificación y Objetivos

La importancia del análisis de datos no atañe solo a grandes empresas. Incluso aquellas que cuentan con un volumen de datos más modesto, pueden extraer fructíferos beneficios de emplear estas técnicas analíticas.

La analítica de datos viene creciendo cerca del 9% anual desde el 2012 [1], otorgándole unos ingresos de 88.000 millones de dólares en EE. UU. y 33.900 millones de dólares en Europa [2].

La inteligencia de negocio o *Business Intelligence*, es definida la mayoría de las veces como *"El proceso de recolección, almacenamiento y análisis de los datos, los cuales permiten proporcionar información para el proceso de toma de decisión de las organizaciones"* [1].

Es por esto que las organizaciones se centran cada vez más en la ventaja estratégica del *Business Intelligence* y en el retorno de la inversión a largo plazo que proporciona. Tanto es así, que cerca del 48% califica la importancia del *Business Intelligence* en la nube como crítica [3].

No obstante, son pocas las herramientas que ofrecen funcionalidades orientadas principalmente a usuarios inexpertos y, en la actualidad, muy pocos de los negocios pequeños perciben los beneficios del análisis de datos. Apenas un tercio de los ingresos generados por la analítica de datos, vienen de pequeñas y medianas empresas [3].

Es, de esa carencia detectada en el sector, de donde nace la motivación de desarrollar este proyecto como Trabajo de Fin de Grado para la titulación del Grado en Ingeniería Informática.

Se pretende desarrollar e implementar un proyecto fácil de usar y gratuito para aquellas empresas que no tengan grandes necesidades analíticas, ni un gran presupuesto, pudiendo realizar ellos mismos las labores de análisis de datos.

Desde este TFG se propone una solución que, primero, ofrece un entorno de analítica de datos colaborativo entre múltiples usuarios, poniendo el foco en aquellos usuarios que no requieran grandes necesidades analíticas.

Y, segundo, ofrecer de forma independiente las librerías desarrolladas, que permiten usar esta página web como si se tratase de un entorno de ventanas similar a los S.O. de escritorio, potenciando que los usuarios estén familiarizados con la Web App, incluso antes de acceder por primera vez.

Este segundo punto consigue que sea muy fácil de mantener y que su acoplamiento sea mínimo, dejando disponible esta interfaz para otro tipo de proyectos.

El objetivo es demostrar mis habilidades en el campo de la analítica de datos y desarrollo web, y a su vez, como ya se menciona, proporcionar una solución para que cualquier negocio pueda introducirse y descubrir los beneficios del *BA*[1] y *BI*[2].

[1] Business Analytics
[2] Business Intelligence

Agradecimientos

Para mí, es un placer, poder usar este espacio para agradecer, a mi familia y a quienes me han acompañado durante toda la titulación, su apoyo constante e incondicional. Teniendo que soportarme en los momentos más complicados de estos últimos cuatro años, haciendo que merezca la pena invertir tanto esfuerzo en el grado.

Por otra parte, querría acordarme de todos los compañeros, que, además de tener siempre un carácter abierto y amable, se han molestado en compartir y explicar su trabajo, y, sus conocimientos de forma desinteresada. Dando facilidades, y aportando soluciones, entendiendo, que el grado no es, ni debería ser, una competición.

Por supuesto, a todos los profesores que a lo largo de mi recorrido académico no solo me han enseñado, sino, que también, me han motivado. Entendiendo, que la dificultad no va estrechamente ligada al aprendizaje.

Por último, a quienes me han guiado durante este trabajo fin de grado, Ana Lavalle López y Alejandro Mate Morga, que han hecho posible y viable este proyecto.

A todos ellos, gracias.

“Ya no estamos en la era de la información. Estamos en la era de la gestión de la información.”

Chris Hardwick 1980, actor.

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de código

1. Introducción

Resulta innegable el protagonismo que están tomando los datos en los últimos años. Las empresas cada vez son más conscientes de este fenómeno y aumentan las inversiones en dicho sector. El último año el sector de la analítica de datos alcanzó una inversión de 134.989 millones de euros [2]. No obstante, las pymes, que suponen el 47% del PIB nacional [4] y suponen el 99,8% de la empresas españolas [4], no se suman con tanta facilidad a la tendencia del *BI*.

Desde aquí mantengo la hipótesis de que dos de los factores que suponen una importante barrera de entrada son la dificultad que encuentra el personal al intentar sacar gráficos por simples que sean -en muchos programas al inicio la línea de aprendizaje es lenta-, y el alto coste que supone a pequeños negocios adquirir estas soluciones de software que permiten abordar estos retos analíticos.

Existen *papers*, que apuntan en la misma dirección. El artículo desarrollado por Ricardo P. Medina Chicaiza, Lorena del C. Chiliquinga Vejar y Amalia P. Ortiz Barba sostiene que "Las razones que explican este divorcio entre esfuerzos y resultados es posible achacarlas a la escasa coherencia entre los objetivos, los instrumentos, los programas y los presupuestos asignados." [5].

Como se menciona en el siguiente fragmento extraído de un artículo de la revista norteamericana Forbes se dice lo siguiente:

En las empresas, la falta de habilidades en el análisis de datos es, para cerca del 27% de los profesionales encuestados, el motivo por el que no se llevan a cabo proyectos de *BI* [7].

A continuación, se adjunta una tabla de precios con algunas de las soluciones más famosas a problemas de *BI*, y el esfuerzo (tanto monetario como en conocimientos) que supone implementar soluciones basadas en su software.

Nombre de la Solución	**Dificultad**	**Precio anual por usuario/servidor en dólares**
Tableau Desktop Professional.	Medio [9]	$840 [7]
Oracle BI	Medio-Bajo [11]	$1.800 [10]
Qlik View Enterprise	Alto	$8.675 [8]
Power BI	Fácil [12]	$120 [9]

Tabla 1. Soluciones BI

Ante esta situación, las pymes son incapaces de afrontar el gasto que supone el uso de software facilitado por la mayoría de soluciones propietarias. Además, para las soluciones *Open Source*, el

nivel de dominio que se requiere hace que sea imposible explotarlas por parte del personal de una pequeña empresa que no dispone de empleados especializados.

Para realizar el proyecto, como más adelante se detallará en el punto 3 de este mismo documento, se ha decidido maximizar el número de posibles usuarios. Por eso, además de ofrecer una plataforma en la que realizar análisis de datos de la forma más sencilla posible, también desarrollamos unas librerías empleadas para nuestra plataforma de análisis de datos, que cualquier usuario con independencia del propósito de su página web, puede incorporar a su plataforma web.

Con el objetivo de conseguir que nuestra aplicación web de analítica de datos resulte extremadamente simple de usar, hemos creado un entorno con características o propiedades muy similares a las que marcan los estándares no escritos que existen en los programas de escritorio convencionales (clic derecho, minimizar o maximizar la ventana, o las opciones de la aplicación en una lista horizontal colocada en el menú superior).

Adicionalmente, desde nuestra plataforma de análisis, al usuario le generará el gráfico con mayor idoneidad al conjunto de datos aportado es aquí donde vamos a basar el grueso de nuestro TFG y el potencial del mismo.

Obteniendo, de esta forma, una puntuación de los tipos de gráficos que resulten más adecuados en función de los datos a representar, si nuestro algoritmo genera poca diferencia de puntos entre varios tipos de gráficos, le ofreceremos al usuario poder escoger entre estos gráficos cual prefiere, haciendo uso incluso de internet para ofrecer un desquite.

Empleamos las APIs de Google, Twitter y Bing para obtener el tipo de gráfico que podríamos calificar como “más de moda”, esto no exonera a los usuarios de poder seleccionar el tipo de gráfico que prefieran y sean ellos quienes tengan la última palabra.

2. Marco teórico

El marco teórico en el que se basa este Trabajo Final de Grado es el estudio denominado « Goal-Based Selection of Visual Representations for Big Data Analytics» realizado por *Matteo Golfarelli, Tommaso Pirini, Stefano Rizzi* [7].

Este estudio discierne y marca las variables para dictaminar que modelo de gráfico es más adecuado para cada conjunto de datos.

A continuación, se puede apreciar, en la tabla de adecuación de visualizaciones de datos, los criterios establecidos para considerar que gráficos son más apropiado frente a otros.

Considerando la siguiente relación de los valores, apto>aceptable>desaconsejado>no apto.

	A) Gráfico de columnas apiladas	**B) Gráfico de una línea**	**C) Gráfico de una línea con marcadores**	**D) Gráfico circular**
OBJETIVO VISUALIZA-CIÓN				
Composición	apto	no apto	no apto	Apto
Orden	no apto	desaconsejado	no apto	no apto
Relación	desaconsejado	no apto	no apto	no apto
Comparación	apto	no apto	no apto	no apto
Clúster	no apto	no apto	no apto	no apto
Distribuición	aceptable	aceptable	aceptable	no apto
Tendencia	desaconsejado	apto	Apto	no apto
Geoespacial	no apto	no apto	no apto	no apto
INTERACCIÓN				
Visión General	aceptable	apto	Apto	apto
Enfoque	aceptable	aceptable	aceptable	no apto
Filtro	desaconsejado	desaconsejado	desaconsejado	aceptable
Detalles bajo demanda	desaconsejado	aceptable	Apto	aceptable
USUARIO				
No experto	apto	apto	Apto	apto
Experto	apto	apto	Apto	aceptable
DIMENSIONALIDAD				
1-dimensional	no apto	no apto	no apto	no apto

2- dimensional	desaconsejado	apto	Apto	apto
n- dimensional	apto	no apto	no apto	no apto
CARDINALIDAD				
Baja	apto	aceptable	Apto	apto
Alta	desaconsejado	apto	desaconsejado	desaconsejado
TIPO DE DATOS INDEPENDIENTE				
Nominal	apto	no apto	no apto	apto
Ordinal	apto	desaconsejado	desaconsejado	aceptable
Intervalo	desaconsejado	apto	Apto	desaconsejado
Ratio	desaconsejado	apto	Apto	desaconsejado
TIPO DE DATOS DEPENDIENTE				
Nominal	no apto	no apto	no apto	no apto
Ordinal	desaconsejado	no apto	no apto	no apto
Intervalo	apto	apto	Apto	desaconsejado
Ratio	apto	apto	Apto	apto

Tabla 2. Adecuación gráfico

	E) Gráfico de burbuja	**F) Gráfico de columnas agrupadas**	**G) Mapa de calor**	**H) Dendrograma**
OBJETIVO VISUALIZACIÓN				
Composición	desaconsejado	aceptable	no apto	aceptable
Orden	no apto	aceptable	no apto	desaconsejado
Relación	Apto	desaconsejado	no apto	aceptable
Comparación	Apto	apto	aceptable	desaconsejado
Clúster	aceptable	aceptable	aceptable	apto
Distribuición	Apto	aceptable	apto	desaconsejado
Tendencia	aceptable	apto	no apto	no apto
Geoespacial	desaconsejado	no apto	apto	no apto
INTERACCIÓN				
Visión General	apto	apto	apto	apto
Enfoque	aceptable	no apto	apto	apto
Filtro	desaconsejado	aceptable	aceptable	aceptable

Detalles bajo demanda	aceptable	aceptable	aceptable	aceptable
USUARIO				
No experto	aceptable	apto	aceptable	aceptable
Experto	apto	apto	apto	apto
DIMENSIONALIDAD				
1-dimensional	no apto	no apto	no apto	no apto
2- dimensional	no apto	no apto	no apto	no apto
n- dimensional	apto	apto	apto	no apto
CARDINALIDAD				
Baja	aceptable	apto	aceptable	apto
Alta	desaconsejado	desaconsejado	apto	aceptable
TIPO DE DATOS INDE-PENDIENTE				
Nominal	no apto	apto	aceptable	apto
Ordinal	desaconsejado	apto	aceptable	desaconsejado
Intervalo	apto	aceptable	apto	desaconsejado
Ratio	apto	aceptable	apto	no apto
TIPO DE DATOS DE-PENDIENTE				
Nominal	apto	no apto	no apto	no apto
Ordinal	apto	no apto	Desaconse-jado	no apto
Intervalo	aceptable	Desacon-sejado	Apto	aceptable
Ratio	apto	apto	Apto	apto

Tabla 3. Adecuación gráfico

	I) Treemap	**J) Gráfico de co-lumnas**	**K) Gráfico de líneas múltiples**
OBJETIVO VISUALIZA-CIÓN			
Composición	aceptable	no apto	no apto
Orden	no apto	apto	no apto
Relación	apto	no apto	desaconsejado
Comparación	aceptable	apto	acceptable
Clúster	apto	no apto	no apto
Distribuición	desaconsejado	aceptable	aceptable

Tendencia	no apto	apto	apto
Geoespacial	no apto	no apto	no apto
INTERACCIÓN			
Visión General	apto	apto	apto
Enfoque	aceptable	aceptable	aceptable
Filtro	aceptable	aceptable	aceptable
Detalles bajo demanda	aceptable	aceptable	aceptable
USUARIO			
No experto	desaconsejado	apto	apto
Experto	apto	apto	apto
DIMENSIONALIDAD			
1- dimensional	no apto	no apto	no apto
2- dimensional	no apto	apto	no apto
n- dimensional	aceptable	no apto	apto
CARDINALIDAD			
Baja	apto	apto	apto
Alta	aceptable	Desaconsejado	aceptable
TIPO DE DATOS INDE-PENDIENTE			
Nominal	apto	apto	apto
Ordinal	desaconsejado	apto	apto
Intervalo	desaconsejado	aceptable	apto
Ratio	no apto	aceptable	apto
TIPO DE DATOS DE-PENDIENTE			
Nominal	Apto	no apto	no apto
Ordinal	desaconsejado	no apto	no apto
Intervalo	desaconsejado	Desaconse-jado	apto
Ratio	Apto	apto	apto

Tabla 4. Adecuación gráfico

Las variables a considerar para nuestro algoritmo y generar una puntuación para ver qué tipo de gráfico se adapta mejor a las necesidades del usuario, son las siguientes, Composición, Orden, Relación, Comparación, Clúster, Distribución, Tendencia, Geoespacial, Visión general, Enfoque, Filtro, Detalles bajo demanda, No experto, Experto, 1-dimensión, 2-dimensiones, ordinal, Intervalo, n-dimensiones, Cardinalidad Baja, Cardinalidad Alta, Nominal y Ratio.

Pasamos a describir cada una de estas variables y, a continuación, mostrar la visualización esperada para cada uno de los tipos de gráficos, contemplados en nuestro proyecto.

Objetivo de la visualización.

El objetivo de la visualización, que se pretende establecer en esta variable, es la finalidad para la que el usuario desea realizar la gráfica.

Este objetivo, puede definirse respondiendo a unas sencillas preguntas mostradas en la *Figura 1*, facilitada por Ana Lavalle López y Alejandro Mate Morga.

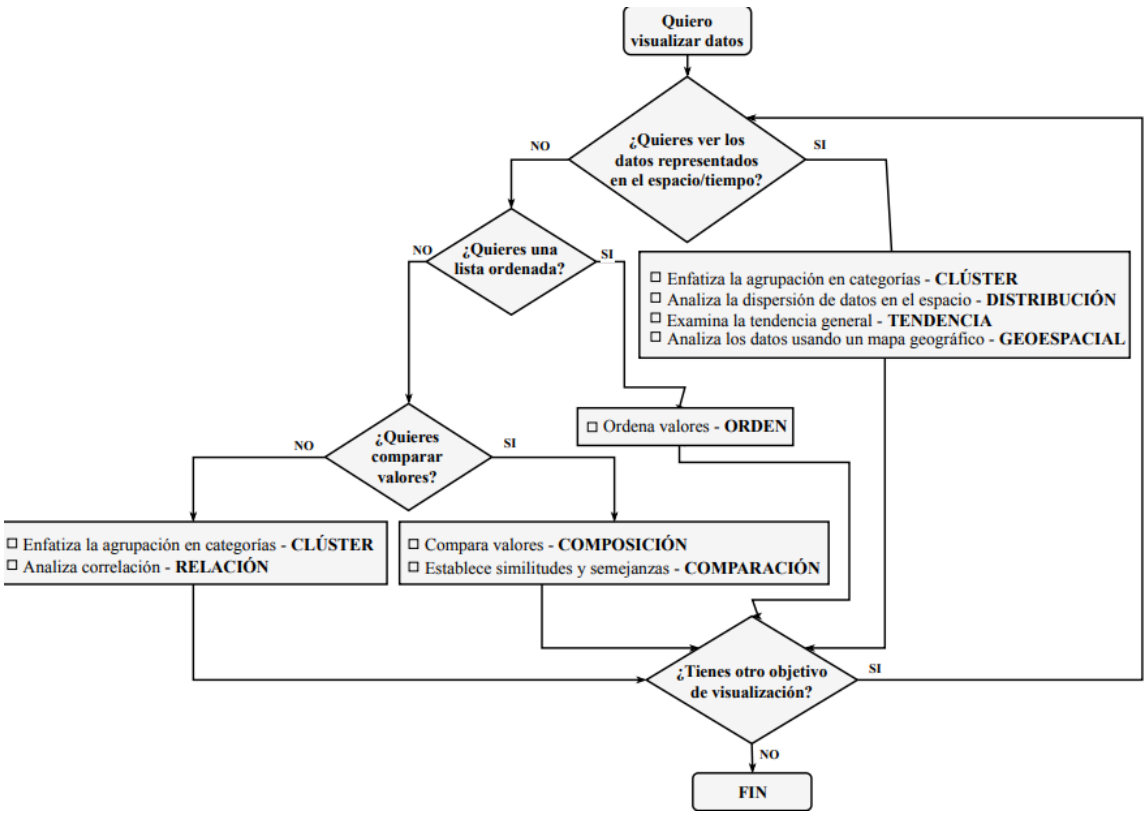


Figura 1. FlowChart. Objetivo de visualización

Interacción.

Esta variable establece como se prevé que el usuario termine interactuando con la gráfica, estas posibilidades son Visión General, Enfoque, Filtro y Detalles bajo demanda

Usuario

El nivel de dominio del usuario con análisis de datos, mostrando gráficas más o menos difícil de comprender, este valor puede ser o Experto o No experto.

Dimensionalidad

Es el número de columnas a visualizar; ante pocas columnas puede ser más conveniente emplear gráficos más simples en los que ver más rápidamente la correlación entre las dimensiones, mientras que si aumentamos estas dimensiones debemos adaptar dichos datos a otro tipo de gráficos.

Cardinalidad

La cardinalidad de los datos, o el número de registros que disponemos, puede alterar la recomendación del gráfico que se haga. Considerando baja cardinalidad un número inferior a 5 y como alta 5 o más registros.

Tipo de datos dependiente/independiente

Es el tipo de dato a identificar en cada uno de los ejes (Y para los dependientes y X para los independientes). El tipo de dato, que, como los propios posibles valores indican, puede ser Nominal, Ordinal, Intervalo o Ratio.

Los gráficos que valoramos para su representación son los vistos en las tablas, cuya representación visual corresponde con la siguiente.

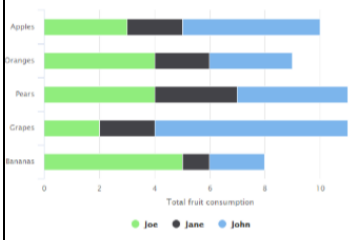
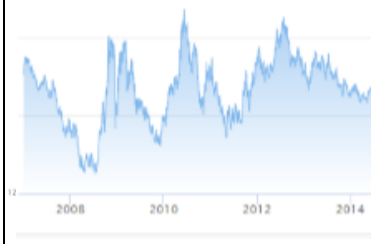
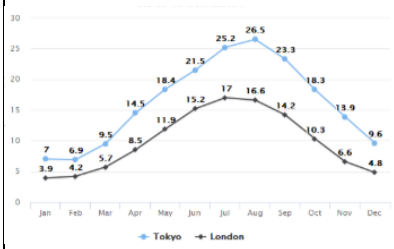
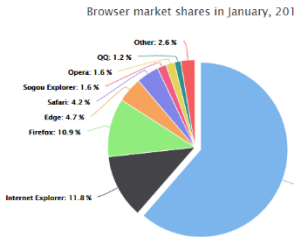
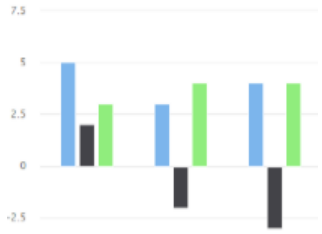
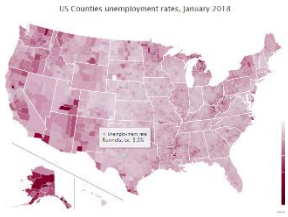
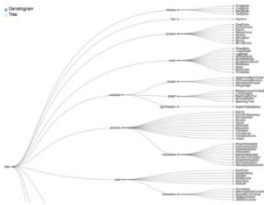
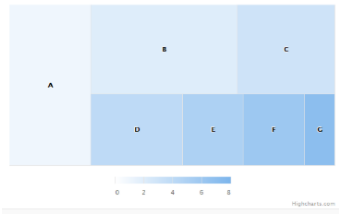
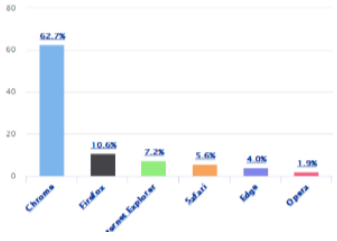
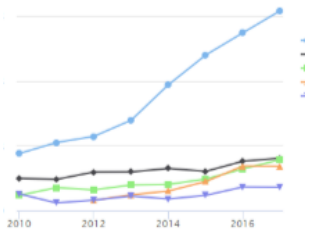
a) Gráfico de columnas apiladas. 	b) Gráfico de una línea 	c) Gráfico de una línea con marcadores 
d) Gráfico circular 	e) Gráfico de burbujas 	f) Gráfico de columnas agrupadas 
g) Mapa de calor 	h) Dendrograma 	i) TreeMap 
j) Gráfico de columnas 	K) Gráfico de líneas múltiples 	

Tabla 5. Representación visual gráficos

3. Objetivos

Los objetivos que este proyecto persigue, es que, cualquier usuario sin conocimientos en el uso de *Business Intelligence*, o *Big Data*, mediante una plataforma web pueda crear y modificar dashboards profesionales que aporten valor a sus organizaciones, pudiéndose beneficiar de las oportunidades que ofrece el *Bussines Intelligence*, antes de invertir grandes cantidades de dinero.

No obstante, esta solución cuenta con carencias importantes que tanto las organizaciones como el personal especializado, con necesidades analíticas más avanzados, echarán en falta.

Las limitaciones con las que contamos, y por lo que posteriormente se aconseja migrar a otras soluciones comerciales, se basan en:

1. La capacidad de procesamiento de los datos se ve influida por la capacidad de los navegadores, el servidor o nuestro propio código.

   Pues para un gran número de registros, las limitaciones de procesamiento de nuestro propio servidor, el navegador o incluso una ausencia de optimización en el código de este proyecto pueden desencadenar fallos considerables.

2. No contar con capacidad para modelar los datos desde el propio entorno.

3. La existencia de ausencias importantes en la representación visual de los datos, pues de cara al usuario -y con motivo de hacerla más sencilla- se ocultan opciones de la librería de gráficos que sí dispondríamos con una implementación de la librería directa en el código (sin nuestro intermediario).

Aun considerando todo esto, creo que la opción de llevar esta propuesta a un uso real es bastante viable.

Primero, porque usamos tecnologías altamente aceptadas por la comunidad, por no decir que son casi un estándar (JS, JQuery y PHP), sin instalación de AngularJS, NodeJS, Laravel o cualquier otro *framework* que imposibilitaría la instalación de la propuesta es un servidor compartido (*Low Cost*).

Con una librería de gráficos (posteriormente hablaremos de ella) que, aunque de pago para su uso comercial, es fácilmente reemplazable por otra solución, sin la necesidad de requerir de *frameworks* adicionales.

Segundo, porque una vez desarrollada la propuesta, es suficiente con publicar nuestro código en un servidor compartido, qué por 40 euros al año, aproximadamente se puede poseer.

Y tercero, es fácilmente escalable a mayor demanda de usuarios, reducimos el coste por usuario.

Este proyecto pone el foco especialmente en los negocios pequeños, que no pueden invertir una gran cantidad de dinero en adquirir software para el análisis de datos, en externalizar el análisis de datos, o contratar personal cualificado.

A nuestros usuarios se les proporciona un entorno gratuito, fácil de usar, y con algoritmos que permite delegar en el software determinadas decisiones, que de forma habitual realiza los analistas de datos. Permitiendo, que las pequeñas empresas sean capaces de emplear técnicas analíticas para sus empresas sin altos costes monetarios.

Como hemos visto, para las empresas medianas y grandes, el software de nuestro proyecto les permite, aunque con limitaciones importantes, descubrir, las bondades del análisis de datos antes de realizar un desembolso importante externalizando el servicio, adquiriendo software nuevo, o contratando más personal.

4. Metodología y herramientas

La metodología que se ha empleado en el proyecto es la conocida como metodología de prototipo, que hace hincapié, en generar prototipos para poder recibir *feedback*, del resultado del producto.

Esta metodología evolutiva, nos permite reutilizar el código empleado para el prototipo, nos ayuda a mejorar la toma de decisiones sobre que funcionalidades incorporar y como incorporarlas. Sobre los prototipos validados, podemos valorar que funcionalidades nuevas podemos incorporar.

Esta metodología, es especialmente útil cuando no se tienen claras muchas de las necesidades del producto final para el rol del cliente, permitiendo recibir un *feedback* de las características del producto. Además de mi propia opinión, se ha consultado a mis tutores del TFG, a dos informáticos sin conocimientos en el área de *Business Intelligence*, y a dos personas ajenas al sector tecnológico.

La periodicidad de dichas reuniones o consultas, tanto telemáticas como presenciales, ha sido proporcional al avance en el desarrollo del código.

Puesto que, para este proyecto, hemos podido consultar las nuevas características del producto, prácticamente cada vez que lo hemos deseado, no nos hemos vistos obligados a desechar grandes cantidades de código.

No obstante, sí que puedo detectar unas líneas generales que se me fueron indicando para ir corrigiendo y modificando el proyecto. Que pasaban por hacer la aplicación lo más intuitiva posible, adaptarla a nuevos usuarios, o incorporar un tutorial, entre otras.

A continuación, se muestran dos imágenes, una del primer “prototipo” y otra de la versión final del proyecto.

Figura 2. Primer prototipo del producto

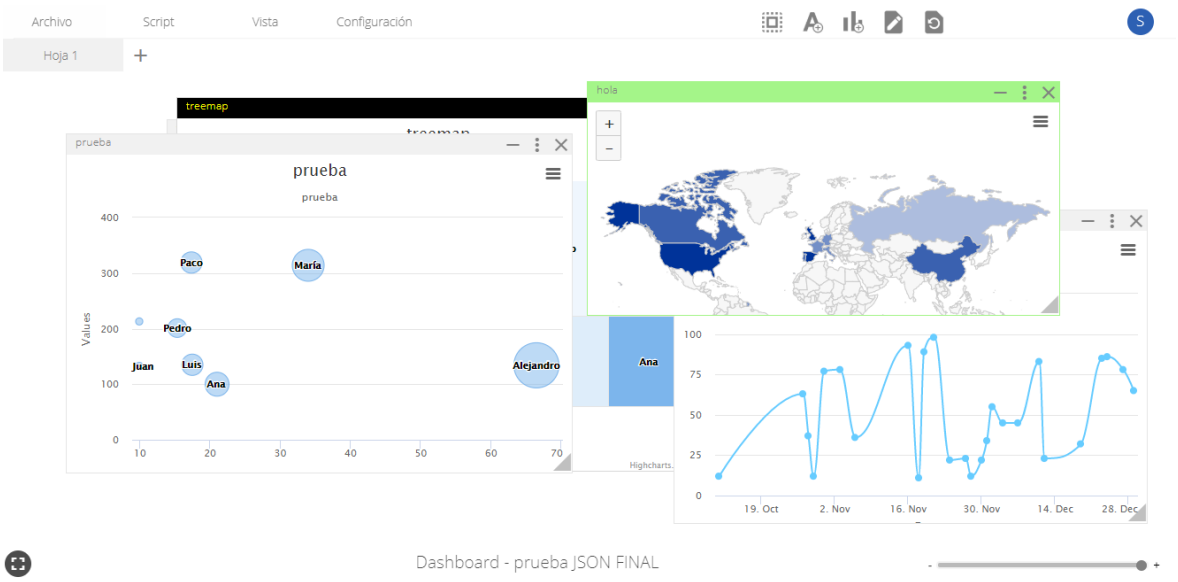


Figura 3. Versión final del producto.

Las herramientas utilizadas, tanto para el desarrollo del Trabajo de Fin de Grado, como las herramientas necesarias para el funcionamiento, son las que se enumeran a continuación.

4.1. Herramientas de desarrollo

Para poder desarrollar el proyecto, hemos usado las herramientas que aquí se nombran.

4.1.1. Notepad++

Como software de desarrollo usamos Notepad++, debido a que tanto PHP como HTML y CSS, no necesita ser compilado para su ejecución, he optado por un cliente ligero y funcional, que suple con creces mis necesidades de desarrollo.

4.1.2. Google Chrome 76.0.3809.100

Empleamos Google Chrome para visualizar el aspecto de la App web durante el desarrollo, y depurar los fallos que nos vamos encontrando durante el desarrollo del proyecto, en todo el proceso de desarrollo, como en el proceso de pruebas, se garantiza un correcto funcionamiento con la versión 76.0.3809.100 para Windows del navegador Google Chrome.

4.1.3. GitHub

GitHub nos ha servido para compartir nuestro código con los tutores de TFG, para que pudiesen estar actualizados de todos los cambios que se van realizando durante el transcurso del proyecto.

Adicionalmente, nos sirve para tener una copia de seguridad en la nube y, en caso de ser necesario, poder revertir los cambios de nuestro proyecto a un punto del pasado.

4.1.4. Trello

Esta herramienta resulta especialmente útil para planificar el tiempo restante de desarrollo aproximado que queda, marcando una estimación de tiempo a cada tarea. Visualizar rápidamente cuantas funcionalidades por hacer quedan, y disponer de una organización de cuáles son esas funcionalidades que deseamos incorporar a nuestra *Web App*.

4.2. Herramientas de implementación

Para la implementación del proyecto se hace uso de las tecnologías que se muestran a continuación.

4.2.1. HTML, CSS, JS, PHP y MySQL

Las tecnologías para el desarrollo de esta WEB son HTML, JavaScript, CSS, PHP y por la parte de base de datos MySQL, las tecnologías de *frontend* son prácticamente estándares de las que escasean alternativas que les puedan hacer frente [6].

El lenguaje usado por la parte del servidor es PHP. El motivo de su elección es que se trata de un lenguaje de programación completamente abierto, y cuyos entornos de desarrollo permiten una rápida y fácil configuración, además es uno de los lenguajes de *backend* más usados, haciendo que reduzca el hipotético coste de contratar a nuevos desarrolladores en este lenguaje.

Empleamos MySQL, debido a que es multiplataforma y a que tiene un menor coste que otras soluciones como Oracle o Microsoft SQL Server.

4.2.2. HTML2Canvas

Usamos la librería HTML2Canvas.Es una librería en javascript que nos sirve para poder generar en PNG el contenido de nuestro dashboard. Para esto ocultamos los elementos que no deseamos que salga en la imagen (como la *header* o el *footer* de la página). Este fichero PNG es generado íntegramente en javascript.

4.2.3. JQuery 3.3.3

Al día de hoy, no es necesario discutir las ventajas que aporta JQuery a los proyectos que lo incorporan. De todas formas, sí queremos remarcar estas características.

Gracias al uso de JQuery podemos reducir considerablemente las líneas de código del proyecto, disminuyendo su dificultad de desarrollo, y haciéndolo más legible.

Por el contrario, necesitamos que el proyecto cargue 3,3kb más en cada vista web, puesto que no supone una diferencia considerable, y que nuestra aplicación web solo tiene 4 vistas, y está pensada

para permanecer en la misma vista la mayor parte del tiempo, no consideramos estos problemas lo suficientemente grandes como para evitar su uso.

4.2.4. D3JS

Aunque no empleemos la librería de D3JS para la visualización de los datos, sí que empleamos alguna de sus opciones para poder trabajar y agrupar los datos en función de las variables independientes facilitadas por el usuario.

4.2.5. HighCharts

Tras realizar el análisis de librerías expuesto en el punto 5.1, nos decantamos por HighCharts. Esta es la opción que cuenta con más gráficos, nos ofrece unas transiciones más completas, se puede emplear para un uso no comercial gratuitamente, y, además es muy fácil de implementar al no depender de librerías ajenas.

Figura 4. Ejemplo de gráficos de la librería HighCharts

No obstante, debemos de tener en cuenta que, a pesar de ser la opción con mayor número de gráficos, hay algunos tipos tales como el dendograma que no están soportados. Por ello se requeriría combinar una segunda librería de gráficos. No obstante, esto se deja planteado para una iteración posterior.

5. Librerías y APIs requeridas

Durante el desarrollo de este proyecto se ha pretendido aumentar al máximo el número de potenciales usuarios. Para ello, hemos decidido desacoplar de nuestra aplicación, las partes de código destinadas a hacer que el proyecto se comporte como un sistema de ventanas y, por otro lado, las funcionalidades que restringen al usuario acceder al código desarrollado para la plataforma.

Por ello, hemos desarrollado dos librerías complementarias para nuestra interfaz Web, la primera, nos permite disfrutar de un entorno de ventanas que se puede utilizar en cualquier proyecto web con JQuery, y la segunda, que proporciona un acceso seguro a la información de la web.

Más adelante, entraremos en profundidad en las características que nos proporcionan ambas librerías.

Adicionalmente, para el correcto funcionamiento del proyecto, optamos por incluir JQuery en su versión 3.4.1, pues nos permite ahorrar costes en el desarrollo del producto, tanto para ambas librerías, como para la interfaz final del proyecto.

A todo esto, debemos sumar que hemos pretendido que nuestro algoritmo de cálculo de puntuación para gráficos funcione como una API, reforzando nuestra idea de que cada una de las partes de este TFG puedan funcionar por separado y de manera independiente, maximizando el número de potenciales usuarios.

5.1. Herramientas de gráficos

Como se ha comentado, hemos decidido decantarnos por HighCharts con la ayuda de D3.js para la agrupación de datos. Esta decisión se ha tomado teniendo en cuenta las siguientes comparaciones.

Al margen de las librerías que a continuación se muestran, se han examinado otras librerías como FusionCharts (buena solución, pero de pago), EJSCharts, NVD3, C3.js, D3 (gran cantidad de gráficos y alta personalización), Statpedia o Vis.js, que no aportan nada nuevo a las comentadas explícitamente en el siguiente análisis.

A falta de una implementación real de los gráficos aplicándolos a grandes cantidades de datos, para poder observar su rendimiento y tiempo de carga, observamos que hay 2 gráficos que destacan sobre el resto; particularmente por las siguientes características, son gratis, no dependen de librerías externas, disponen de una gran cantidad de tipos de gráficos a escoger, son interactivos con el cursor del ratón, teóricamente su implementación no es difícil y son visualmente agradables.

Las librerías que cumplen con estas condiciones son, AmCharts y HightCharts. Lo que ha sido decisivo para decantarnos por Highcharts, es que teóricamente los tiempos de carga son considerablemente mejores con respecto a AmCharts [7].

5.1.1. Librería Chartis

Sitio web: www.gionkunz.github.io

Implementación.

Como en el caso anterior, y como veremos en el resto de librerías, la forma de usar la librería es insertar código JavaScript que modifique la etiqueta asociada al fichero HTML, la forma de insertar los datos es mediante un array de arrays.

Ventajas.

Como sucede en el caso anterior vemos que es fácil de implementar y es gratis, salvo que, en este caso, no dependemos de ninguna librería o *framework* adicional de JavaScript, la cantidad de parámetros para personalización es decente y considerable, aunque tampoco son excesivos. En este caso, podemos indicar el tamaño del gráfico bajo unas determinadas dimensiones de pantalla.

Desventajas.

El número de gráficos que dispone está incluso más limitado que en el caso de n3charts, y, por defecto, no interactúa con el cursor para mostrar más información, dicho de otro modo, se comporta como una fotografía estática sin poder interactuar con él

5.1.2. Librería AnyChart

Sitio web: www.anychart.com

Implementación.

La forma en la que insertamos la configuración del gráfico es mediante JS, reflejándose en la etiqueta asociada del fichero JS. Como en el caso de Gionkunz, para determinados gráficos, los datos son cargados mediante un array de arrays.

Ventajas.

Las opciones de personalización del gráfico son mayores a las de los casos anteriores, cuenta con gran cantidad de gráficos, mapas, cajas, e incluso multitud de gráficos combinados, es *responsive*, y tampoco requiere de uso de librerías o *frameworks* adicionales, por último, cuenta con varios estilos predefinidos desde la propia web, para poder realizar pruebas de colores en línea.

Desventajas.

Aunque es bonito, no es de los que más destaque por su aspecto visual. Pero, sobre todo, el gran problema es que es de pago, aunque cuente con gráficos de pruebas sin limitaciones, según informa la web oficial de Anychart. “Todas las versiones de prueba son completamente funcionales y no tienen límite de tiempo para que usted pueda evaluar cómodamente todo el potencial de los productos de AnyChart.”.

5.1.3. Librería Zingchart

Sitio web: www.zingchart.com

Implementación.

Su implementación se basa en insertar código JavaScript, en el cual podemos modificar los parámetros del gráfico, reflejándolo en la etiqueta del fichero HTML, la forma de insertar los datos es mediante un *array* de JSON, en el que cada JSON incorpora un array de arrays.

Ventajas.

Es *responsive* y cuenta con una enorme cantidad de gráficos; tampoco necesitamos librerías o frameworks externos, y posee una fácil implementación.Por otro lado, aunque cuente con una cantidad de opciones de personalización justas, son suficientes. Podemos seleccionar rangos en las gráficas de líneas para profundizar más en un determinado periodo de tiempo.

Desventajas.

No es especialmente bonito, por la experiencia dada en la asignatura INGP donde incluíamos el gráfico de palabras con esta librería, hay que mencionar que no era especialmente veloz en comparación con Google Charts.

5.1.4. Librería AmChart

Sitio web: www.amcharts.com

Implementación.

Como en la gran mayoría de casos vistos, configuramos el gráfico mediante JS, y se muestra en la etiqueta asociada del fichero HTML.

Ventajas.

Sin necesidad de indicarlo explícitamente los gráficos son *responsive* por defecto. Conforme pasamos el cursor o seleccionamos un rango nos muestra más información, tampoco requiere de librerías o *framewroks* de terceros. Gran cantidad de gráficos -incluyendo mapas-, gráficos combinados muy completos, con un diseño muy bonito y altamente personalizable.

Desventajas.

Quizás lo único que se podría llegar a encontrar problemas es en los tiempos de carga, o en la poca información que una vez empecemos encontremos, a priori, y tras buscar por internet, nada hace pensar que esto pueda suceder.

5.1.5. Librería GoogleCharts

Sitio web: www.developers.google.com/chart/

Implementación.

La forma en la que insertamos la configuración del gráfico es mediante JS, reflejándose en la etiqueta asociada del fichero JS. Como en el caso de Gionkunz, para determinados gráficos, los datos son cargados mediante un array de arrays.

Ventajas.

Cuenta con una gran cantidad de gráficos, es completamente gratuita y no se requieren dependencias externas; además cuenta con el soporte de Google. Su implementación es muy sencilla.

Desventajas.

No es feo, pero tampoco bonito. Los controles para profundizar en una determinada zona del gráfico, no es muy intuitivo pues se hace con un slider situado en la zona inferior, tampoco es *responsive*, por defecto.

5.1.6. Librería HighCharts

Sitio web: www.highcharts.com

Implementación.

Su implementación se basa en, al igual que en los anteriores, insertar código JavaScript que modifique la etiqueta asociada en el fichero HTML para dibujar el gráfico. Los datos son procesados en un array de JSONS.

Ventajas.

Extremadamente similar al anterior, con gran cantidad de gráficos (incluso mayor), tampoco requiere de librerías externas, no obstante, y en función por lo recopilado por internet es especialmente bueno para grandes volúmenes de datos.

Desventajas.

Diseño más funcional que el anterior, con menos transiciones, haciéndolo algo más feo.

5.1.7. Librería recharts

Sitio web: www.recharts.org

Implementación.

A diferencia del resto de gráficos analizados, ReCharts proporciona definir las propiedades de los gráficos mediante etiquetas HTML. Aunque sigue siendo imprescindible el uso de JS, para poder generar el gráfico de forma dinámica.

Ventajas.

Gráficos muy bonitos, gran parte de la implementación (como ya hemos mencionado) se realiza mediante etiquetas HTML, y bastantes tipos de gráficos, en función acercamos el cursor muestra más datos.

Desventajas.

Requiere NodeJS; por defecto no son gráficos *responsive*, además, no podemos profundizar en una sección concreta en el tiempo en un gráfico de líneas.

5.1.8. Librería n3-charts

Sitio web: www.n3-charts.github.io

Implementación.

Su implementación se basa en insertar código JavaScript que modifique la etiqueta asociada en el fichero HTML correspondiente, para dibujar el gráfico. Al igual, que hace la mayoría de las herramientas webs mostradas en esta memoria. Sus parámetros de configuración son también en JS, bajo el $scope propio de AngularJS.

Ventajas.

Fácil implementación y que es gratis, se ajusta por defecto (sin indicar nada en la configuración de JS) al ancho de la ventana, además, el gráfico muestra más información conforme interactúa con el cursor del ratón.

Desventajas.

Funcionan bajo angular, y tienen una cantidad de gráficos muy, muy, limitada; a esto hay que sumarle un diseño no especialmente bonito y que no puedas seleccionar un periodo concreto del gráfico de líneas para examinarlo más exhaustamente.

5.2. Ficheros adicionales, sistema de escritorio

Como hemos mencionado, la primera librería, a la que denominaremos LFE -librerías con funcionalidades de escritorio-, es de elaboración propia y sirve, como su propio nombre indica, para poder emplear un entorno de ventanas, similar y con características parecidas a la que nos ofrecen los programas de escritorio, pero desde el navegador web. Permitiendo que sea muy sencilla su implementación, para que aquellos usuarios sin capacidades altas de desarrollo web puedan incorporar nuevas funcionalidades, o bien para que programadores con más experiencia no inviertan su tiempo en generar estas características.

El motivo por el que se ha decidido separar en un fichero independiente, lo relacionado con esta librería es la ausencia de una librería con funcionalidades similares a las que buscaba y a la creencia que dicho desarrollo puede servir a terceros a desarrollar sus propias aplicaciones.

Para poder emplear esta librería deberemos incluir en nuestro documento HTML las siguientes referencias.

```
<script src="../js/jquery.min.js"></script>
<script async defer src="../js/desktopWebLib/script.js"></script>
<script async defer src="../js/miscript.js"></script>
```

Código fuente 1. Dependencias del proyecto.

Cabe destacar que cualquier atributo adicional o modificación del contenido HTML se realiza empleando la nomenclatura y recomendaciones del W3C en la que indica, por ejemplo, que cualquier atributo propio debe empezar por `data-` [8].

En la librería LFE que se ha desarrollado expresamente para este TFG, disponemos de las propiedades siguientes.

Apilar barras, en cualquiera de los 4 laterales del navegador, para ello hacemos uso de las clases de CSS `.bar.top`, `.bar.bottom`, `.bar.left` y `.bar.right`.

Es posible apilar más de un elemento, como podemos observar en la siguiente imagen.

Figura 5. Apilación de barras

Sobre esta barra superior, por ejemplo, podemos incorporarle menús, cuyo uso también está recogido por la librería, haciendo posible el uso de menús y sus submenús de manera sencilla.

Para ello debemos implementar el siguiente código.

```
<ul class="list-element-horizontal">
  <li class="dropdown">
    Menú
    <div class="options" id="file_menu_options">
      <ul class="list-menu">
```

```
        <li>
          <a href="#">SubMenú</a>
          <ul>
            <li >SubSubMenú</li>
          </ul>
        </li>
      </ul>
    </div>
  </li>
</ul>
```

Código fuente 2. Creación de un menú.

Observemos el resultado asociado a este fragmento de código.

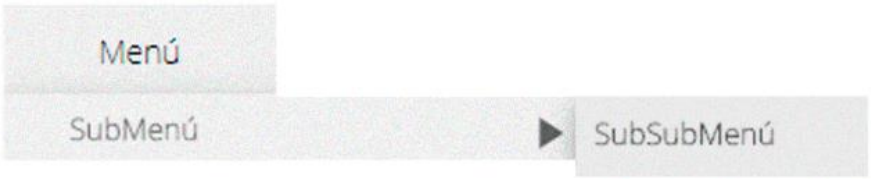


Figura 6. Menú desplegable.

Podemos ver cómo somos capaces de desplegar submenús y más submenús asociados, incluso añadiendo una fecha sobre el elemento en cuestión para indicar al usuario que se trata de otro submenú.

También se incorpora la posibilidad de mostrar información adicional al mantener el cursor encima, pudiendo modificar el estilo de visualización fácilmente mediante CSS en la clase `.info_element`.

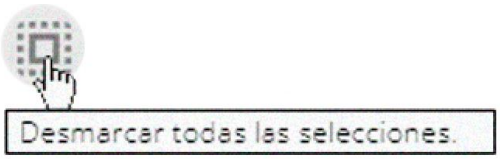


Figura 7. Superposición del cursor en elementos

Para ello empleamos la siguiente etiqueta.

```
data-hover-info="Desmarcar todas las selecciones."
```

Código fuente 3. Uso de la superposición del cursor

También incorporamos el uso de las hojas, que permite cambiar entre múltiples pestañas, como sucede en programas conocidos como Microsoft Excel, Chrome, o Adobe Photoshop.

Asimismo, añadimos ventanas, para que de forma considerablemente fácil se pueda disponer de un sistema de ventanas, y las opciones normalmente asociadas a estas, mover objeto (en los ejes X, Y y Z), redimensionar, minimizar o maximizar.

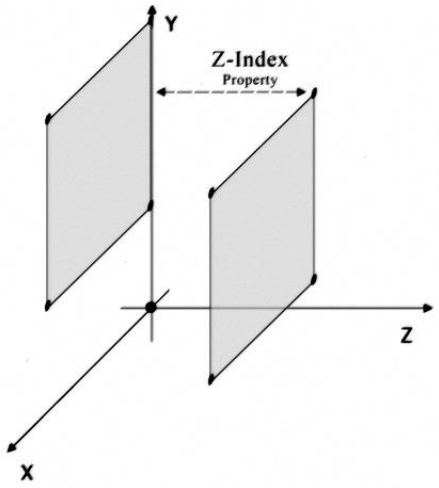


Figura 8. Profundidad eje Z en CSS

Para mover un objeto en el eje Z (profundidad), empleamos la propiedad de CSS `z-index`, no obstante, esto ocasiona conflictos con el resto de los elementos de la interfaz que siempre deben permanecer superpuestos a cualquier objeto. Las barras superiores, los menús, el menú contextual o los modales son ejemplos de estos objetos que se deben mostrar por encima de todo, por ello usamos el siguiente rango de valores.

Elemento	**Valor z-index**
Las Ventanas	De 0-500
Los elementos de los Modales	800 a 900

Elementos Modal especial de confirme de cierre	1500 a 1600
Elementos del marco de la aplicación web	1000 a 1100
Menú contextual	550
Tutorial	2000

Tabla 6. Estipulación profundidad de los elementos

A este movimiento libre de las ventanas, se le suma que pueda ser redimensionada de forma libre; para ello además se permite que las ventanas puedan ocupar media pantalla arrastrando la ventana a uno de los extremos izquierdo o derecho del navegador web.

Por último, vamos a mencionar de forma escueta como implementar algunas de las opciones incorporadas.

1. **Modales**

Para desplegar un modal que se solape a todos los elementos existentes en la web y extraiga el contenido de dicho modal de un fichero externo se emplea el siguiente atributo.

```
data-modal-page="./modals/new_text.html"
```

Código fuente 4. Mostrar un modal.

También se permite superponer modales sobre otros por orden de aparición, si queremos que un botón cierre el modal que contiene dicho elemento le añadiremos el atributo `data-close-modal`, si además queremos que nos pida confirmación antes de cerrar el modal incorporaremos el atributo `data-confirm-close="5569"` siendo 5569 el identificador de la ventana del modal correspondiente.

2. **Paneles.**

Si queremos que se pueda avanzar y retroceder con los botones *next* y *back* respectivamente entre varias vistas.

Deberemos primero marcar las vistas que deseamos que cambien al presionar los botones con el atributo `data-panel="diseño".`

Para el botones siguiente que al situarnos en la última vista accionará unas determinadas funciones en JS definidas por `data-finished="insertWindowChart,closeModal"`.

Para conseguir que avance o retroceda, debemos equipar al elemento HTML con las clases `.before` y `.next`.

Figura 9. Botones de before y next

3. Menú contextual

Si queremos que al realizar clic derecho sobre un elemento se muestre un menú contextual personalizado, deberemos añadir dentro del html el siguiente fragmento de código.

```
<div class="option_click_right">
   <ul class="hover">
      <li data-function-dismis-window>Minimizar</li>
      <li data-function-close-window>Quitar</li>
      <hr>
      <li data-function-config-window>Configuración</li>
      <li>Imprimir</li>

      <li data-down-windows>Bajar una posición</li>
      <li data-up-windows>Subir una posición</li>

      <hr>
      <li data-send-to-front>Enviar al Frente</li>
      <li data-send-to-back>Enviar al Fondo</li>
   </ul>
</div>
```

Código fuente 5. Implementación menú contextual

4. Sección de código.

Para escribir código en el navegador y, por tanto, se creen unas líneas para teclear fragmentos de código en las que a la izquierda se visualice la numeración de las líneas y el color de las líneas cambie en función de si estamos en una línea par o impar tendremos que incorporar el siguiente fragmento html.

```
<div contenteditable="true" id="script_edit">
</div>
```

Código fuente 6. Inclusión del editor de script.

En la siguiente fotografía podemos observar el resultado.

Figura 10. Editor de Script

5. Alertas de error personalizados.

Desde Javascript podemos mandar mensajes de error con la llamada a la función `mostrarAlertError("Título","Mensaje");` , cuyos parámetros son el título del mensaje y el contenido del mensaje a mostrar en dicho panel.

Figura 11. Mensaje de error.

6. Selecciones

Para que sea simple poder seleccionar varias ventanas en nuestro dashboard lo que procedemos a hacer es añadir el atributo `data-selectionable`, de esta forma permitimos que al hacer clic cambie el color superior de la barra y podamos seleccionar otros elementos con la ayuda de la tecla `Ctrl`.

7. Opciones de Deshacer y rehacer

Para estas ventanas también se ha implementado la posibilidad de rehacer o deshacer un cambio, para ello se deberá presionar la tecla `Ctr+z` o `Ctr+Shift+Z`, o bien incorporar la clase `undo` y `redo` respectivamente en el elemento html que queremos que sirva para deshacer.

8. Data-color

Por último, una opción menor es la de apilar colores en un mismo elemento con ayuda de los atributos `data-color-id="1" data-color="#f1f1f1, white, grey,black, grey"` donde indicamos el id del elemento de color (ese valor no puede coincidir con otro `data-color-id`) y los colores a representar.

Figura 12. Listado de colores.

5.3. Librería de seguridad en el lado del Cliente

Además, hemos generado, un fichero js adicional con el que añadir seguridad a nuestra plataforma web, protegiendo nuestro código en el lado del cliente, en caso de ser necesario. Para ello:

1. Bloqueamos las teclas que comprometen la integridad de los datos como las que despliegan el panel de desarrollador, `Ctr+C` o `Ctrl+X`, o incluso `impPt`.

Bloquear la tecla `ImpPt` supone complejidades técnicas diferentes, puesto que desde el navegador, por motivos de seguridad no se puede acceder al portapapeles, para modificarlo, deberemos de mostrar un input de tipo texto cuyo alto y ancho sea de `0px`, cuya opacidad sea de cero, y que se encuentre al fondo de cualquier elemento, con un `z-index` negativo, añadiendo el falso input en el código, insertamos el texto que queremos que se copie cuando se presione `impPt` y copiamos el contenido del input, remplazando la captura realizada, por el contenido de dicho input.

2. Bloqueamos la Herramientas para desarrolladores, incluso, cuando se accede desde el menú del navegador, esto es no es simple, pues no basta con controlar el `Ctr+Shift+I`, sino debemos tener un temporizador para saber cuánto tiempo lleva abierta la consola del navegador.

3. Bloquear el menú contextual al hacer clic derecho.

4. Aunque esta característica sea opcional, también, podemos ocultar la pantalla cuando esta pierde el foco, protegiendo los datos y la información ante un intento de usar el programa captura de Windows.

Figura 13. Se difumina el contenido al perder el foco

Esta configuración la determinamos en el `body` de nuestro fichero html, aquí se muestra un ejemplo de la etiqueta a insertar.

```
<body data-security="true">
```

Código fuente 7. Activar opciones de seguridad

Y para incluir estas funcionalidades es tan fácil como incorporar la librería en el head del fichero.

```
<script async defer src="../js/security.js"></script>
```

Código fuente 8. Incluir la librería en el head.

6. Interfaz de usuario

Con lo referente a la interfaz propia que hemos desarrollado para este proyecto, tenemos que decir que se ofrece un paquete de opciones que pasamos a describir.

Ante todo, mencionar, que el diseño web utilizado para este desarrollo se basa como idea principal en los mockups que se mostrarán a continuación.

Que reciben inspiración en la plataforma colaborativa de ofimática Google Docs, y de la plataforma de *Bussines Intelligence*, QlikView.

Al no disponer de muchas pantallas para navegar dentro del proyecto, tampoco he tenido la necesidad de crear una gran cantidad de Mockups.

Algunos de los mockups más importantes diseñados para este proyecto son los siguientes.

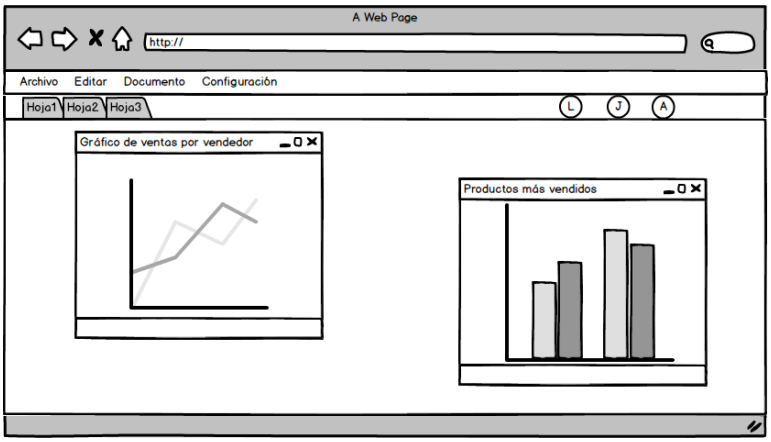


Figura 14. Mockup dashboard.html

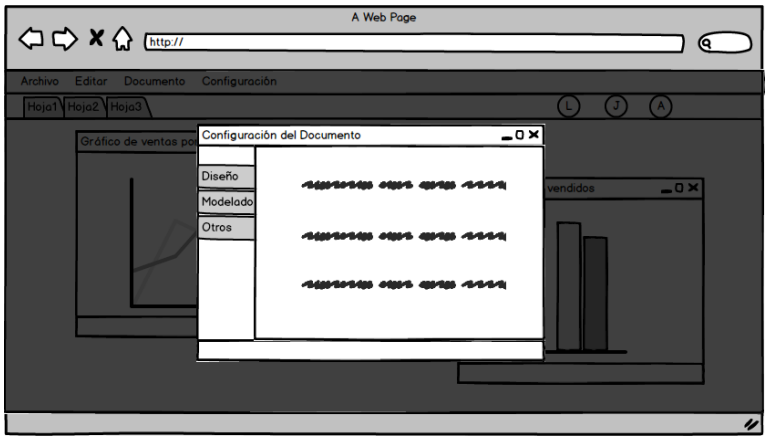


Figura 15. Mockup configuración ventana

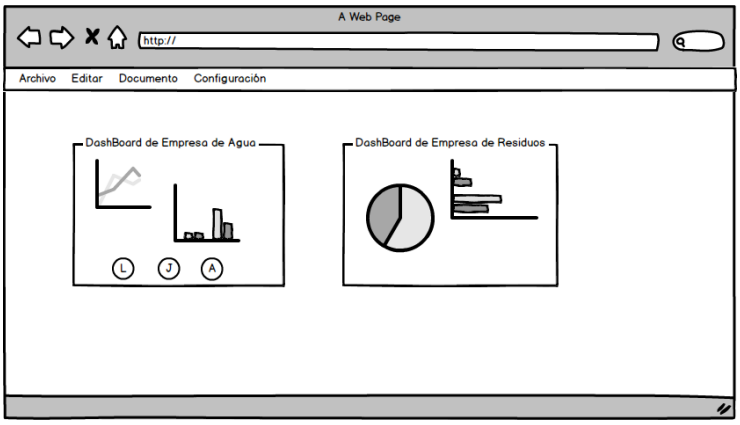


Figura 16. Mockup panel de selección de dashbaord

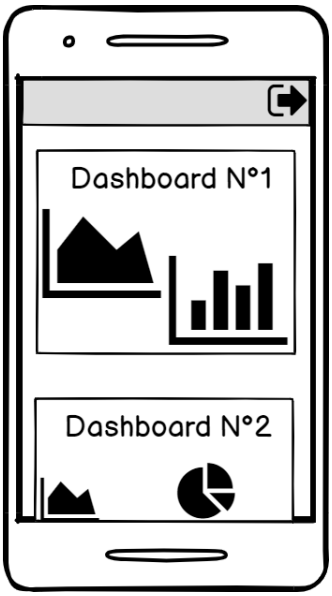


Figura 18. Mockup móvil. panel de selección de dashboard

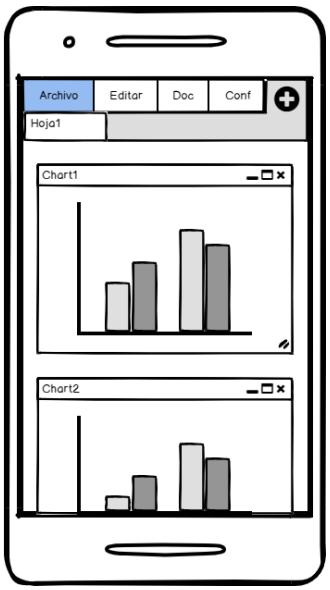


Figura 17. Mockup móvil, configuración ventana

Como ya comentábamos, en la realización del proyecto, se ha optado por desarrollar librerías complementarias independientes entre sí, provocando que el código sea más fácil de modificar y mantener en el tiempo. Por supuesto, permitiendo a terceros emplear estas librerías para el desarrollo de páginas webs ajenas a este proyecto.

Por esto mismo, muchas de las cosas que se comentan a continuación se encuentran en el uso de las herramientas facilitadas en las librerías ya descritas, y en este punto del documento se omitirá su implementación. Pero no la forma en la que hemos decidido implementar nuestro LFE.

Por ello, a continuación, se describirán todas las características de la interfaz del proyecto, las funcionalidades incorporadas, la forma de usarlas, pero no su implementación.

Empezamos a describir lo primero que se ve al acceder a la aplicación. Debido al carácter y enfoque que tiene nuestra aplicación para usuarios inexpertos, la primera vez que accedamos mostraremos un tutorial -que en cualquier momento podemos saltar- donde se visualizarán 4 vídeos con las funciones más relevantes de nuestro proyecto.

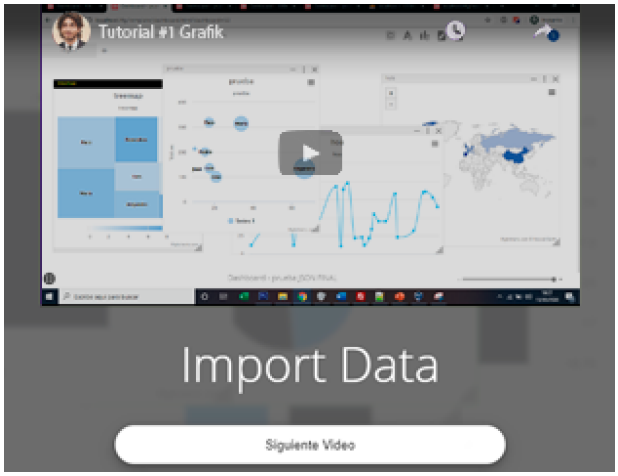


Figura 19. Video del tutorial

En cualquier momento, podemos volver a visualizar el tutorial desde *Configuration→Show tutorial*.

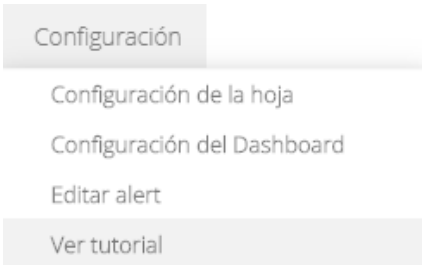


Figura 20. Acceder al tutorial desde el menú superior

Antes de acceder al tutorial, dispondremos de un *login*, cuya interfaz pretende ser muy simple e intuitiva, buscando que el usuario en todo momento se sienta cómodo, facilitando lo máximo la curva de aprendizaje y no identifique a la web como una herramienta hostil. Por ello, el *login* ha sido inspirado en la suite de Google, con la que la mayoría de los usuarios ya está familiarizada.

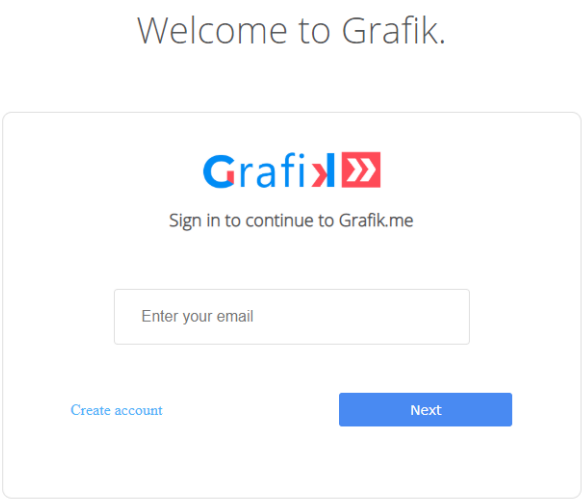


Figura 21. Login de nuestro proyecto

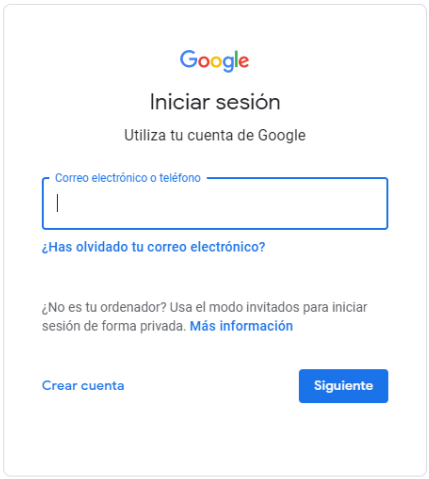


Figura 22. Login de Google

Realizado el *login*, accedemos a un panel donde se visualizan todos los *dashboards* que hemos creado, con una pequeña previsualización de los gráficos del *dashboard*.

Se podrá previsualizar, de forma muy esquemática, los gráficos que posee cada uno de nuestros cuadros de mandos, como podemos apreciar en la siguiente imagen.

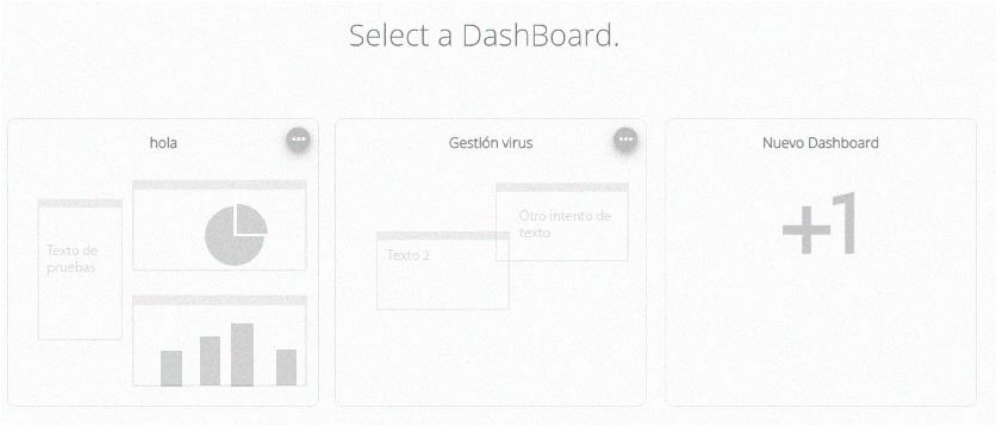


Figura 23. Vista de selección de dashbaord

Para realizar esto, hemos debido deserializar el objeto Javascript guardado en el servidor, y multiplicar sus propiedades de altura, anchura, y coordenadas X e Y por 0.3 para mostrarlo a escala en estas vistas.

Por supuesto, al realizar clic derecho sobre la previsualización de cualquier dashboard nos ofrece una serie de opciones entre las que se encuentran, las opciones de eliminar, descargar, y configuración de ese mismo cuadro de mandos.

Tras clicar en un *dashboard* este se abrirá al completo donde se incorporan la mayoría de las funcionalidades que se ofrecen desde este TFG, y donde el usuario destinará la mayor parte del tiempo.

En este panel, se pueden ver las dos barras superiores, la primera que posee los menús, una zona destinada a acciones rápidas, y los usuarios que tienen acceso a ese *dashboard*.

La segunda barra superior, está destinada a contener todas las hojas que creemos en el documento, la implementación de estas barras es mediante las clases `.bar` y `.superior`.

Este diseño se inspira, en otras interfaces más conocidas como *QlikView*, o *Google Drive*, pero más simplificado con la clara finalidad de hacer que el usuario se sienta más cómodo, en la plataforma.

Figura 24. Barra superior Google Drive

Figura 25. Barra superior QlikVIew

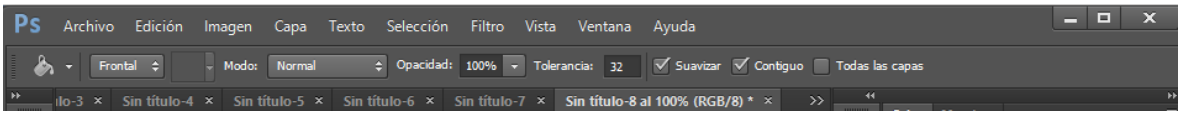


Figura 26. Barra superior Photoshop

En los cuatro menús superiores de nuestra solución encontramos las opciones clasificadas por “File”, “Script”, “View” y “Configuration”.

Figura 27. Menú de nuestra aplicación

En el primer menú, “Archivo”, se engloban las siguientes funcionalidades.

1. Nueva Hoja. Crea otra hoja en el Dashboard principal (segunda barra superior).
2. Añadir Objeto. Contiene un submenú para seleccionar el tipo de objeto a incorporar en la hoja seleccionada, este objeto puede ser de tipo texto o de tipo gráfico.
3. Guardar. Debido a la naturaleza online del proyecto el guardado se realiza de forma automática; no obstante, podemos forzar a realizar el guardado de todo el documento, por si se llega a producir alguna complejidad técnica.
4. Exportar Dashboard. Exportamos nuestro contenido en un formato propio.
5. Importar Dashboard. Importamos un cuadro de mando desde un fichero con extensión propia previamente exportado.
6. Nuevo Dashboard. Desde esta opción creamos un nuevo, cuadro de mandos.

7. Compartir. Obtenemos una URL por la que podemos compartir nuestro cuadro de mandos con otros usuarios para que puedan ver y editar el Dashboard.
8. Deshacer / Rehacer. Deshace/rehace el último movimiento realizado por el usuario, a estas características podemos acceder usando la combinación de teclas `Ctr+Z` o `Ctrl+Shift+Z`.

Del segundo menú, denominado "Script*"* encontramos las siguientes características.

1. Editar Script. Editamos los datos de nuestro modelo, si no queremos subir los datos tal cual y deseamos realizar una pequeña transformación.
2. Recargar. Recargamos los datos con la información que se dispone en el servidor, por si se produce algún bug o desconexión con el servidor, proporciona al usuario la certeza de que los datos son actualizados con la información del servidor.
   Además, cargará de nuevo, todos los datos indicados en la ventana del Script -por si se han realizado modificaciones-, realizar esto constantemente puede no ser siempre beneficioso, pues podríamos añadir carga innecesaria al servidor.

Desde "Vista", únicamente disponemos de un submenú, denominado "Aplicar a todos los gráficos", las opciones del submenú son.

1. Apilar. Moveremos todas las ventanas en bloque a la izquierda o arriba.
2. Alinear. Alinearemos las ventanas en función del eje seleccionado, para que se muestren de forma consecutiva y no se solapen en el eje Y o X.

Y por último del menú de "Configuración".

Donde abrimos los modales de configuración de los siguientes objetos,

1. Configuración del gráfico, hoja o *dashboard*, se despliega un modal con las opciones de cada uno de estos elementos.

Otra propiedad muy importante, que es el elemento principal de nuestro dashboard, y viene implementada por la LFE, es el mencionado sistema de ventanas. Que dispone de las siguientes características.

1. Una barra superior, donde se muestra el título y que al arrastrar desde ese punto movemos la posición de la ventana a lo largo de toda nuestra aplicación.

Dispondremos en dicha barra superior de tres botones, los cuales ejecutan las acciones de eliminar, cerrar, configurar la ventana.

2. También, dispondremos del icono inferior derecho que modifica las dimensiones de la ventana.
3. Un menú contextual, que se activa realizando clic derecho sobre, nuestro objeto y podemos ver las siguientes opciones sobre nuestra ventana, minimizar, cerrar, abrir la configuración de dicha ventana, bajar o subir una posición, o enviar al frente y enviar al fondo.

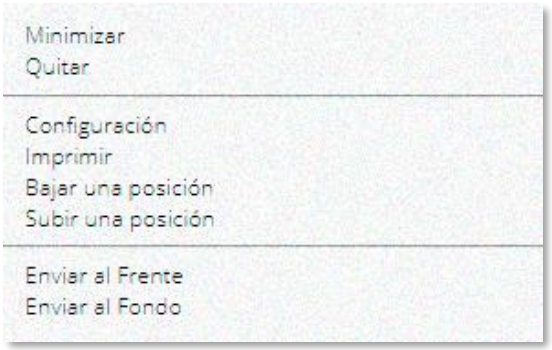


Figura 28. Menú contextual de nuestro objeto

También ofrecemos un cuadro de programación bastante sencillo, en el que poder indicar que ficheros queremos importar a nuestra aplicación web, bien es cierto que dicha importación solo será efectiva al momento de clicar en la opción de refrescar el script .

Como se aprecia en la figura 18, en la zona superior hemos decidido crear tres zonas bien diferenciadas.

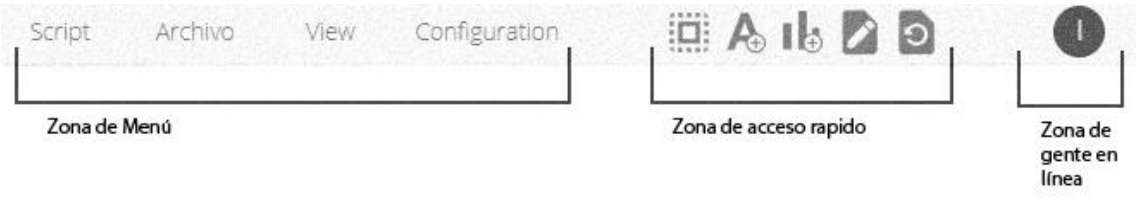


Figura 29. Barra superior de nuestro proyecto

Debido al carácter web de la solución que proporcionamos, se oferta la posibilidad de que se acceda desde cualquier dispositivo, incluyendo aquellos con dimensiones más pequeñas.

En este diseño *responsive*, cuando accedemos a una resolución inferior a 630px, ocultamos la mayoría de las opciones de edición, pues entendemos que el propósito no será otro que la visualización de sus datos y no su modificación.

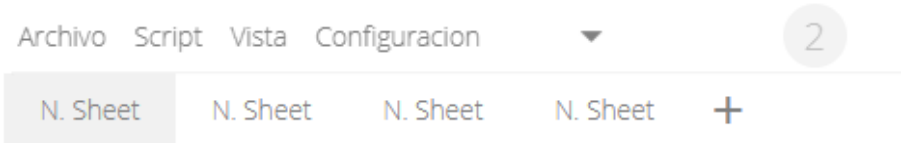


Figura 30. Panel superior responsive

Cuando accedemos a esta resolución pensada para móviles, a lo que se procede es a reorganizar las ventanas, para que ocupen todo el ancho disponible y se apilen unos encima de otros.

En la versión de escritorio, al presionar sobre los perfiles del marco superior a la derecha se despliega una zona que proporciona información del usuario, última vez que se conectó al *dashboard* y todos los cuadros de mandos que tenemos en común con él.

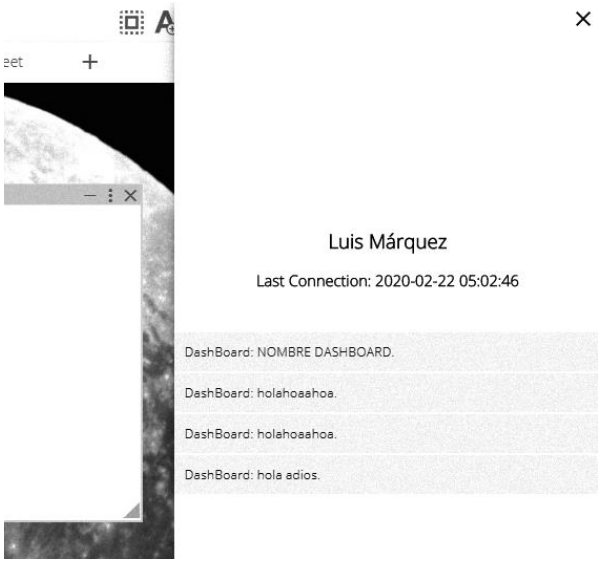


Figura 31. Menú lateral desplegable

6.1. Editor del Script

Proporcionamos un microentorno de programación en el que incluir tres sentencias básicas para procesar de forma muy básica los datos.

Se puede incluir datos desde diferentes fuentes, ficheros de texto plano, JSON de un servidor ajeno, o una base de datos convencional.

Figura 32. Importación de otros ficheros

Las posibles 3 formas de incluir ficheros de datos a nuestro dashboard son la siguiente.

1. Mediante conexión a una base de datos externa.

```
[connectBBDD('UserName,'Password','nameDatabase','table',
mysql://luismarquez.es)]
```

2. Mediante JSON.

```
[connectJSON('https://luismarquez.es/file.json')]
```

3. Mediante un CSV.

```
[connectJSON('https://luismarquez.es/file.csv')]
```

Implementar esto, ha supuesto una dificultad inesperada, especialmente para hacer que el fondo sea dinámico, añadiendo o eliminando líneas.

Pues en un primer momento se barajó la opción de hacerlo con un `textarea` superpuesto al fondo, cuyo número de líneas debe ser comunicadas a nuestra función en JS, para agregar líneas en el fondo por cada salto de línea; pero, tras ejecutarlo, lo vi inviable pues el *scroll* se debía producir entre el fondo y el `textarea` de forma simultánea produciendo *lag*. Al final, nos decantamos por la opción de desarrollar todo esto desde CSS con ayuda de los `contenteditable` y las pseuedoclases de CSS.

Todos estos cambios se guardarán al hacer clic en *Save*.

Pero con el fin de no sobrecargar el servidor, y como ya se mencionó, no será efectivo hasta presionar en *refresh*  .

6.2. Opciones de nuestro perfil

Por supuesto, también podemos configurar toda la información relativa a nuestro perfil, como el correo, el nombre de usuario o la contraseña.

Podremos también activar un modo noche y cambiar el idioma del usuario.

Para ello, debemos ir al botón con nuestra inicial situado justo en la esquina superior derecha de la vista para seleccionar un dashboard.

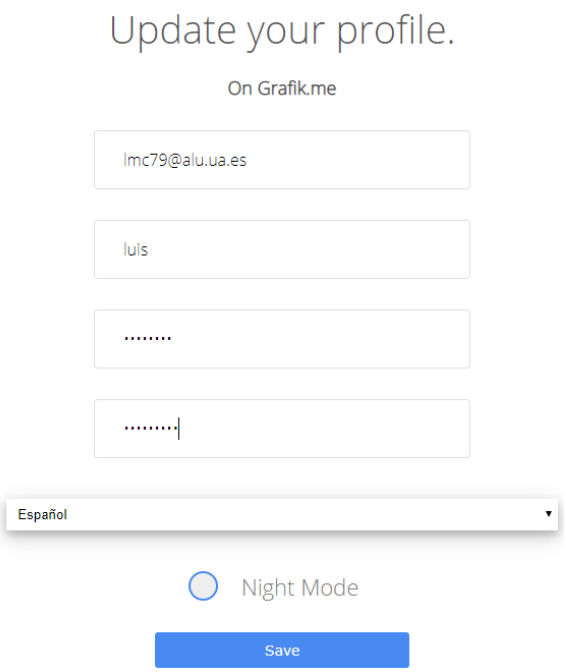


Figura 33. Actualizar perfil

6.3. Opciones del DashBoard

Para el dashboard también podemos modificar una serie de opciones que podríamos clasificar de típicas o standard, como lo son, su título o su imagen de fondo.

No obstante, también se puede modificar el diseño de las ventanas.

Además, podemos cambiar, los estilos generales de los gráficos, entre los siguientes 3 estilos.

Estilo 1.

Figura 34. Barra superior de la ventana con la hoja de estilos 1

Estilo 2.

Figura 35. Barra superior de la ventana con la hoja de estilos 2

Estilo 3.

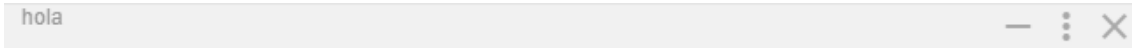


Figura 36. Barra superior de la ventana con la hoja de estilos 3

Aunque similares, estos tres estilos tienen aspectos diferenciadores haciéndolos preferibles por el usuario según sus gustos, y pudiéndolos hacer personalizables modificando el fichero .css ubicado en la carpeta `/css`.

6.4. Opciones de las hojas

En la configuración de la hoja disponemos de 3 campos a modificar, cambiar el nombre por defecto de la hoja, añadir una imagen de fondo, o en caso de no establecer imagen de fondo o no estar disponible poner un color de fondo.

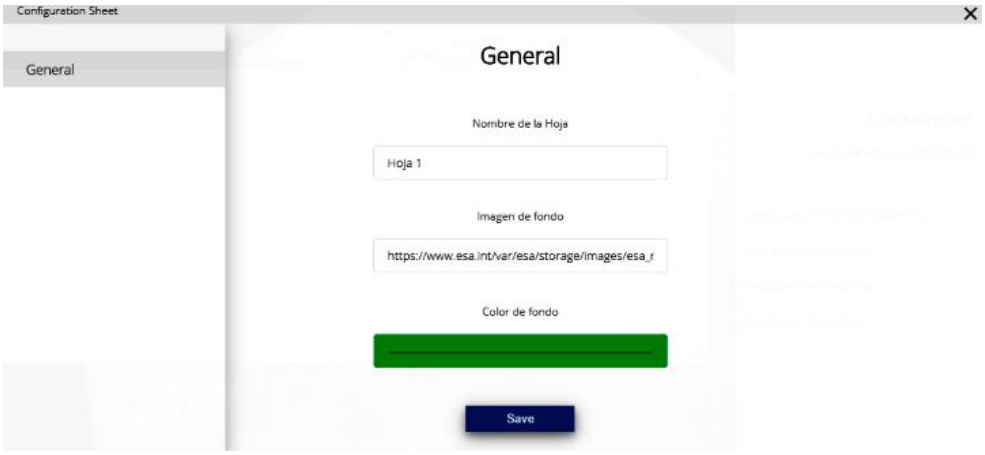


Figura 37. Modal configuración hoja

Hay que decir que la configuración con respecto al fondo de cada hoja tiene prioridad sobre el fondo seleccionado en el dashboard, siendo el fondo del dashboard efectivo cuando no existe una configuración establecida para el fondo de las hojas.

6.5. Opciones de los objetos

Las propiedades comunes de los objetos de texto y de gráficos, son el color de las ventanas, el texto del título y su subtitulo, el código CSS o JS a inyectar, hacer transparente el fondo de las ventanas o eliminar la barra superior de las ventanas.

6.5.1. Opciones en los gráficos

En los gráficos, además, se nos permite cambiar, el tipo de gráfico y modificar su estilo, pero no modificar las variables dependientes e independientes. Aunque si podamos ver que valores fueron seleccionados en la creación del gráfico.

6.5.2. Opciones en el texto

En el texto se proporcionan capacidades de edición relativas a girar el texto, el color y el tamaño.

Si se desea agregar una imagen, basta con indicar la ruta de la imagen como fondo y no introducir texto en el cuadro de mando.

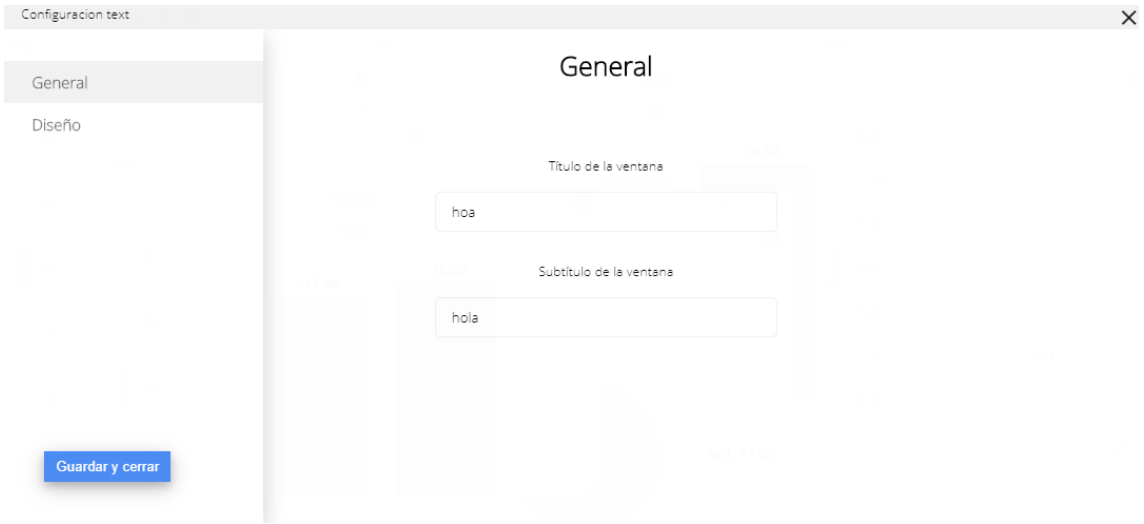


Figura 38. Configuración del texto

6.6. Entorno colaborativo

Para fomentar que varias personas puedan modificar el mismo *dashboard*, se tolera poder trabajar en el mismo *dashboard* simultáneamente sin sobrescribir los cambios, actualizando las propiedades de los gráficos en el servidor de cada valor modificado, y mostrando dichas modificaciones en todos los equipos conectados a ese mismo cuadro de mandos.

Estos cambios se almacenan sin la necesidad de darle a guardar y mostrando constantemente los últimos cambios en todos los dispositivos conectados.

Consultando la información del servidor cada segundo, asegurando que transcurrido ese periodo tenemos todos los datos actualizados del servidor.

A continuación, se muestra en dos ventanas distintas el mismo dashboard.

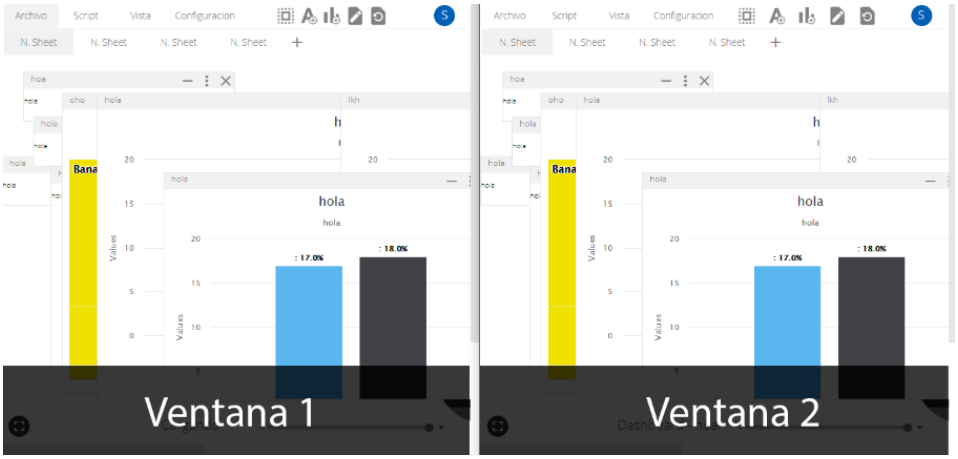


Figura 39. El mismo dashboard abierto en dos ventanas distintas.

Tras modificar los valores de la ventana 1, la ventana 2 se ve actualizada.

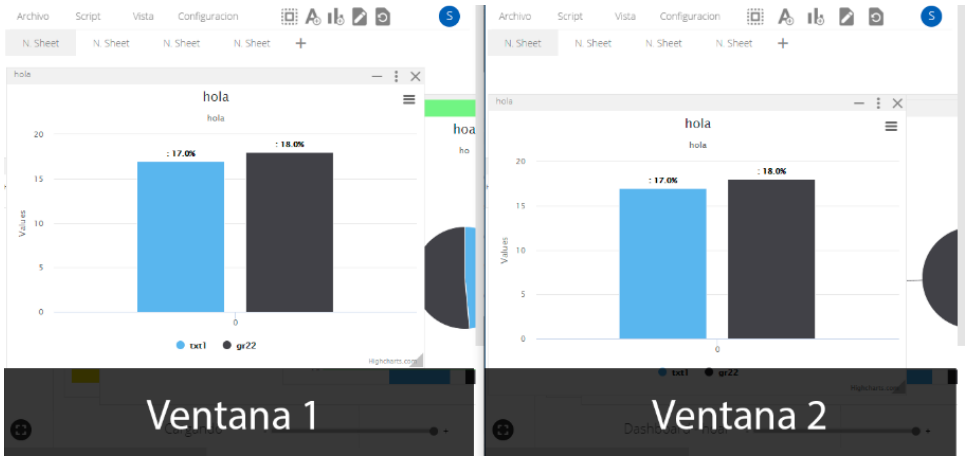


Figura 40. El mismo dashboard modificado en dos ventanas distintas.

7. Desarrollo en el lado del Servidor

7.1. Base de datos

La forma que usamos para almacenar todos los datos referentes a nuestra aplicación es una base de datos relacional MySQL, en la que muchas propiedades de las ventanas, las hojas, o los gráficos, al ser provenientes de objetos en JavaScript, es serializado y almacenado en un único campo.

Otras propiedades tales como `sheetId`, `dashboard`, o `dataID`, sí que disponen de su propio campo específico en la base de datos.

Por otro lado, para almacenar todo el contenido de los datos que se nos ha proporcionado por parámetro para la realización de los gráficos, tanto si viene de una Base de datos, un CSV o un JSON, convertimos toda esa la información a un fichero JSON con las mismas características, almacenado en la ruta `/data` de nuestro proyecto, que posteriormente leemos para generar las gráficas. Guardando en nuestra base de datos únicamente la ruta de este fichero.

La entidad "alert" no se llega a utilizar, pues se pretende aprovechar en futuras mejoras del proyecto.

El entidad-relación queda de la siguiente forma.

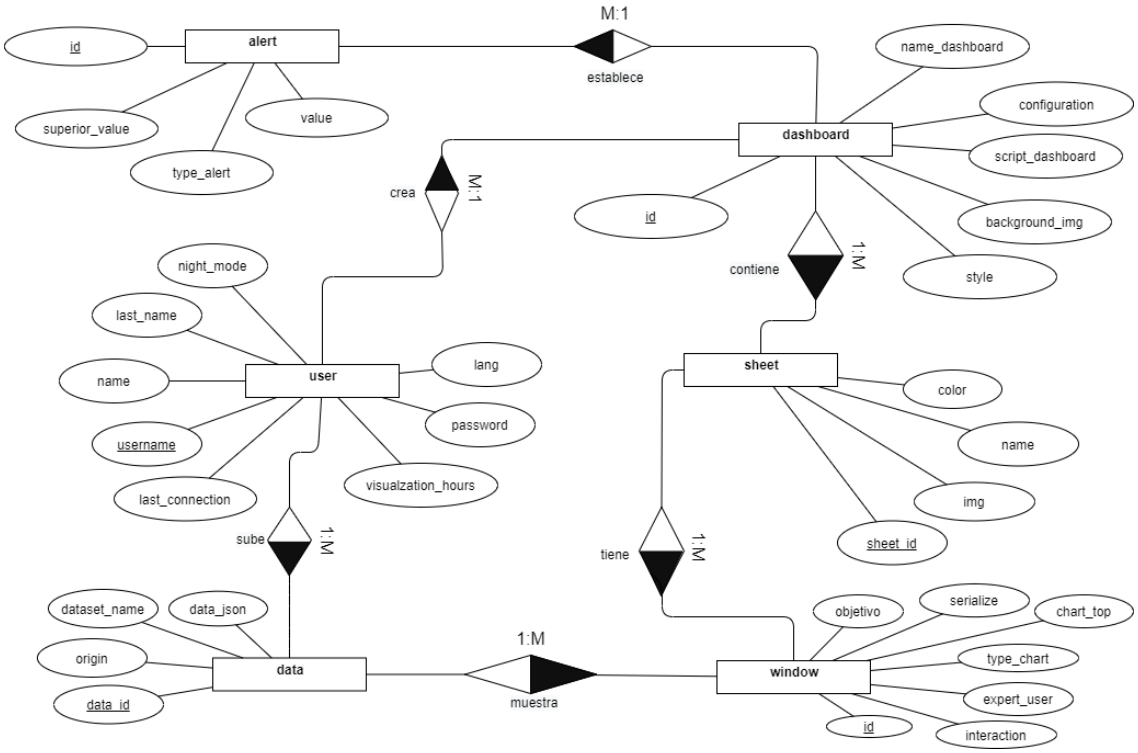


Figura 41. Entidad relación del proyecto

7.2. Fichero database.php

El fichero `database.php` ubicado en la carpeta `/model` compone una serie de funcionalidades usadas por todos los ficheros php, estas funcionalidades son las siguientes.

1. **Conexión a la BBDD.**

La cadena de conexión y las operaciones que realizamos a la base de datos proceden de funciones centralizadas en este fichero. Pudiendo cambiar la forma de conexión fácilmente en caso de ser necesario. También disponemos de `get_last_id()`, obteniendo el id de la última *insert* realizada.

2. **Parámetros de entrada.**

Para los parámetros que se obtiene por url, llamamos a esta función para poder sustituir la forma de capturar las entradas de `Get` a `Post`, modificando la variable `debug_enable`.

3. **Procesado de entrada en la base de datos**

Los parámetros que se terminan insertando en la BBDD deben ser transformados, o bien para evitar inyecciones de código MySQL, o bien para poder almacenar un JSON serializado en javascript; por ello, hacemos uso de la función `real_scape_string()`.

4. **Encriptar y desencriptar**

También disponemos de funciones de encriptar y desencriptar, lo que facilita que si desde cualquier fichero php se desea encriptar esos datos baste con la llamada a la función almacenada en `database.php`. Por ejemplo, al exportar e importar nuestro cuadro de mandos hacemos uso de dichas funciones.

7.3. Procesado de los datos

Como hemos mencionado, los datos que recibimos para generar las gráficas pueden proceder de otra BBDD, ficheros CSV o ficheros JSON.

De los datos hallados deberemos poder encontrar, la dimensionalidad de los datos, su cardinalidad y el tipo de dato (ordinal, cardinal, etc.).

Por ello, los pasos que realizamos son los siguientes.

1. Detectar la procedencia de los datos.
2. Convertir estos datos a un formato JSON.
3. Estipular su cardinalidad contando el número de filas.
4. Indicar la dimensionalidad de los datos.
5. Mirar el tipo de datos que son.

Con todos estos procesos realizados, podemos pasar a emplear nuestro algoritmo de selección de gráfico.

7.4. Algoritmo de elección del gráfico

Para identificar que gráfico se adapta mejor a las necesidades de los datos, hacemos un array bidimensional en el que establecemos valores para cada una de las propiedades del gráfico.

Establecemos los siguientes pesos a cada una de las variables del gráfico, los valores asociados, son:

apto = 1

aceptable = 0.5

desaconsejado = -1

no apto = -100

Como hemos dicho, primero creamos un *array* bidimensional en el que la primera posición corresponde al tipo de variable y la segunda al tipo de gráfico.

Le indicamos que para el gráfico 2 (gráfico de barras) en la variable 0 (objetivo de visualización) tiene una puntuación de -100, pues corresponde con el valor “No apto” de la tabla de idoneidad del punto 2 de este documento.

En el código también se observa la variable peso, no obstante, se ha optado por mantener `$peso` con valor uno en todo el código, considerando que todas las variables son igual de importantes en la selección del gráfico.

```
//  2    //LINEA
//  4    //CIRCULAR
//  10    //COLUMNAS

//  0    //Orden
$arrayVisualization[0][2]=-100*$peso;
$arrayVisualization[0][4]=1*$peso;
$arrayVisualization[0][10]=-100*$peso;
//  1    //Relación
$arrayVisualization[1][2]=-0.5*$peso;
$arrayVisualization[1][4]=-100*$peso;
$arrayVisualization[1][10]=1*$peso;
```

Código fuente 9. Establecimiento de pesos en PHP de los gráficos lineal, circular y de columnas.

El hecho de que algunas variables se le asocie un -100 se debe a que cuando en alguna de nuestras variables obtengamos el valor “No apto”, bajo ningún concepto podamos considerar ese gráfico como recomendado, y debido a que solo trabajamos con 13 variables distintas, jamás podremos superar un 100 de puntuación como para que empiece a dar valores positivos, y considerar esta recomendación.

Una vez establecidas las variables de entrada, realizamos una suma de cada uno de los atributos de cada tipo de gráfico. Para ello, en el primer parámetro del *array* se pone el tipo de gráfico y en el segundo la variable a puntuar, siendo el tipo de gráfico el valor que itera conforme avanza el bucle.

El código de la implementación en el qué ordenamos es una lista con los gráficos con mayor idoneidad.

```
    foreach ($arrayVisualization[0] as $clave => $valor) {
      $json[$cont]->nombre=$nombre[$clave];
      $json[$cont]->keyChart=$clave;

      $json[$cont]-> valueChart= $arrayVisualization[$usua-
rio][$clave]+$arrayVisualization[$objVisualizacion][$clave]+$arrayVisua-
lization[$interaccion][$clave]+$arrayVisualization[$dimensionali-
dad][$clave]+$arrayVisualization[$cardinalidad][$clave]+$arrayVisualiza-
tion[$datodependiente][$clave]+$arrayVisualization[$datoindepen-
diente][$clave];

    }
    usort($json, function($a, $b) {
        return $a->valueChart <=> $b->valueChart;
    });
```

Código fuente 10. Extracción de puntuación de idoneidad

Empleamos otro método para poder ampliar la diferencia entre los gráficos seleccionados por nuestro algoritmo. Este método, que permite ampliar la holgura de puntos, requiere de acceso a internet.

Pues realiza una búsqueda en motores de búsqueda como Google o Bing, obteniendo la relevancia entre los tipos de gráficos consultados.

Para esta parte, obtenemos el número de veces que se repiten los términos en Google o Bing, la cantidad de resultados que se encuentran y las veces que son mencionados en Twitter considerando los *Likes* y *RTs* de cada Tweet, pudiendo obtener la popularidad de estos gráficos.

Para conseguir esto hemos debido de acceder a la API de *Twitter*, y *Google*, además de realizar *Web Scrapping*[3] a los primeros resultados de *Google*, combinando todos estos datos, y una vez normalizados, obtenemos una puntuación para el gráfico en cuestión.

Esto se realiza siempre que empaten en puntos pues la "popularidad" de los gráficos a usar debe ser un factor secundario que no prime sobre la utilidad real del mismo.

[3] Técnica para extraer información de una página web. [22]

8. Instalación

Para la instalación del proyecto, deberemos descomprimir nuestro fichero .zip, formando el siguiente esquema de directorios. Este debe ubicarse en un servidor web con php instalado.

El fichero descomprimido, suma aproximadamente 25 megabytes y más de 15 000 líneas de código.

```
/
      /css/->Ficheros de estilos
            /desktopWebLib/->Los estilos más genéricos asociados a esta
librería
      /fonts/ ->La fuente del texto
            /roboto/->Los ficheros de la fuente roboto
      /template/->Ficheros .html
            /modals/->Ficheros .html que se muestra en los modales
      /export/->Ficheros .grafik creados al exportar un dashbaord
      /imgs/->Imágenes usadas en nuestro proyecto
            /icon/->Imágenes relacionadas con el logo de grafik
            /style/->Fondo para los estilos
            /styles/->Imágenes representativas de cada estilo
      /data/->Fichero de datos subido por nuestros usuarios
      /js/->Archivos .js
            /desktopWebLib/->El código js más genérico asociado a esta li-
breria
            /Highcharts/->Código js para la representación de gráficos
      /model/->Ficheros .php necesarios para la ejecución del proyecto
            /calcPuntuation/->.php para obtener la informaciónde las api
```

En el directorio raíz veremos el fichero `install.php`, el cual, tras abrirlo se nos mostrará un breve asistente el cual nos permitirá crear la base de datos y sus tablas en nuestra base de datos.

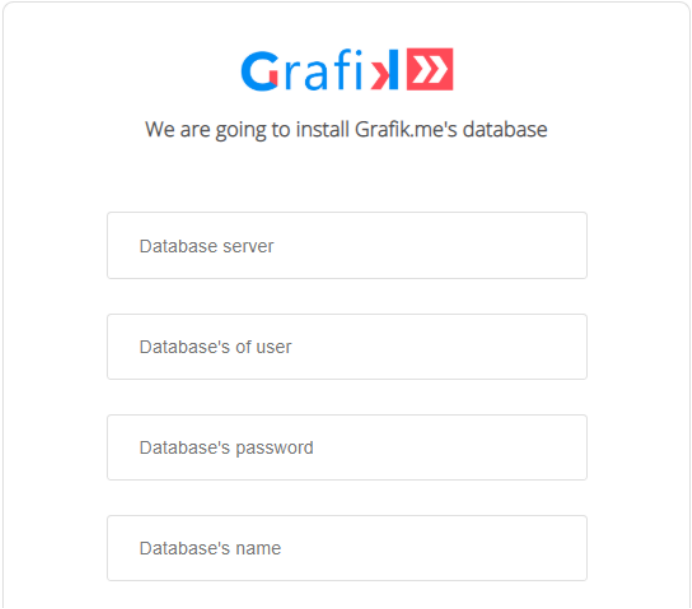


Figura 42. Asistente de instalación BBDD.

Incluyendo la modificación del fichero `/model/database.php` para que cualquier consulta use automáticamente los parámetros de conexión adecuados a la BBDD, haciéndola más fácil de usar para usuarios inexpertos.

9. Experimentación

9.1. Pruebas de rendimiento

Para las pruebas de rendimiento y en la parte del *FrontEnd*, para conocer el tiempo de carga se hace uso de las siguientes 2 herramientas: la primera, Google Speed Test, cuya web proporciona una puntuación con la velocidad de carga de la misma; y la segunda el panel de herramientas de los propios navegadores donde se indica el tiempo de carga de cada uno de los recursos.

Para poder realizar las siguientes pruebas, se ha creado un fichero que genera un *dataset* del volumen deseado con datos que pueden ser representados desde cualquier tipo de gráfico, a continuación, se muestra el código de dicho *script*.

```
$num_regis=1000;

$nombre=["Maria","Pedro","Juan","Alicia","Laura","Paco","José","Raúl"];
$pais=["GBR","CAN","ESP","ITA","FRA","PORT"];
$content="[";

for ($i=0;$i<$num_regis;$i++) {
  if ($i!=0) {
    $content=$content.",";
  }
  $content=$content.'{
    "id":'.$i.',
    "name":"'.$nombre[rand(0, count($nombre)-1)].'",
    "dato": '.rand(5, 40).',
    "dato2": '.rand(100, 200).',
    "countryID":"'.$pais[rand(0, count($pais)-1)].'",
    "date": "'.rand(1, 31).'-'.rand(1, 12).'-'.rand(1970, 2020).'"
  }';

}
$content=$content."]";

file_put_contents('./dataset_out_'.$num_regis.'_generate'.rand(10000,
999999).'.php', $content);
```

Código fuente 11. Código fichero generador de datasets

En las siguientes tablas podemos ver las respuestas a cada una de las herramientas en función del número de información de datos mostrados en dicho *dashboard*.

Estas pruebas las hemos realizado en *Google Chrome* con distintas funciones.

Tiempo de respuesta medio (tras tres ejecuciones) obtenido de las herramientas de desarrollador de Google Chrome, aplicando los registros a las funciones `count`, `average` y `sum`.

Número de gráficas	1 000	10 000	100 000	500 000	1 000 000
Count	8,05	12, 3	8,52	27,63s	50.94
Average	11.75	10,73	9,46	33,82	45.01
Sum	8.91	12,16	9,07	25,7	55,05

Tabla 7. Tabla de pruebas con distinta cantidad de gráficos

Para poder realizar las pruebas correctamente desde Google PageSpeed Insights, hemos debido subir el proyecto a nuestro servidor personal y eliminar el acceso con contraeña, para que esté accesible.

Puntuación obtenida por *Google PageSpeed Insights* sobre 100.

Plantillas HTML	login.html	selectDashboard.html	dashboard.html, 10 mil registros.
GoogleSpeedTest	100	96	37

Tabla 8. Puntuaciones obtenidas en Google Speed Test

Observamos cómo tras la ejecución de *Google PageSpeed Insights*, aquellas páginas que requieran de acceso a los datos, y su descarga, empeora significativamente el resultado obtenido por Google.

No obstante, minimizamos este impacto con el usuario real mediante llamadas asíncronas, mostrando la información en cuanto esté disponible. Esta puntuación es importante para Google, pues se usa para posicionar mejor los enlaces en los resultados de búsquedas, no obstante, nuestra plataforma solo indexará la página de login.html o selectDashboard.html (previo inicio de sesión).

Provocando, que la mala puntuación obtenida para el dashboard.html, no afecte negativamente en la indexación de Google.

9.2. Pruebas de usabilidad

Para las pruebas de usabilidad no podemos usar una herramienta adecuada para estos casos, como userttesting.com, por lo que intentaremos seleccionar a tres personas que no hayan participado en las iteraciones del prototipado del proyecto y describan en voz alta como se desenvuelven en el uso de la plataforma.

Para ello, con los medios que tengo a mi alcance he seleccionado a tres personas con destreza en el uso de software. Sin ser profesionales de la informática, ni del *Business Inteligence*.

A ellos, se les propondrá que estén 5 minutos familiarizándose con el entorno de la web, y posteriormente se les pedirá una serie de acciones, que pasaremos a cronometrar y comparar con los valores que nosotros consideramos óptimos.

Además, se les pedirá que razonen en voz alta todo el proceso para poder extraer conclusiones.

	Esperado.	Tiempo usuario1	Tiempo usuario2	Tiempo usuario3
Crear un nuevo dashboard	< 15 segundos	8 segundos	7 segundos	4 segundos
Compartir un dashboard	< 15 segundos	12 segundos	16 segundos	11 segundos
Importar un fichero de datos	<180 segundos	274 segundos	No conseguido	692 segundos
Crear un gráfico	< 40 segundos	32 segundos	32 segundos	23 segundos
Crear un objeto de texto	< 40 segundos	64 segundos	28 segundos	26 segundos

Tabla 9. Tiempo de respuesta de los usuarios.

De estos resultados, podemos extraer que tenemos funcionalidades que están claramente con un tiempo superior al esperado, mejorar estos aspectos de la usabilidad se dejará para futuras revisiones del proyecto, además nos encontramos con otros aspectos de la usabilidad que están muy por debajo del tiempo esperado.

Los usuarios coinciden en que es recomendable facilitar la importación de los datos, no obstante, ahora mismo para paliar este problema -que se abordará en profundidad próximamente- se ha incluido en el tutorial dos vídeos explicando la forma adecuada para la importación de los datos.

10. Conclusiones

Quiero aprovechar el espacio final de conclusiones de éste TFG, para hacer balance del trabajo realizado, el aprendizaje adquirido y las posibles mejoras que en un futuro se le puedan aplicar a este proyecto.

A lo largo de esta memoria, se ha podido observar que he hecho énfasis en estas cuestiones de minimizar la línea de aprendizaje y reducir lo máximo posible el rechazo que pueda generar el entorno a nuestros usuarios inexpertos, potenciando que cualquier persona inexperta, no solo en el análisis de datos, incluso también en el ámbito tecnológico, pueda sentirse cómoda con la propuesta de este proyecto.

Y, aunque no en todos los usuarios se ha logrado, sí que estoy satisfecho del trabajo logrado en líneas generales.

No obstante, el proyecto cuenta con limitaciones importantes a la hora de personalizar los elementos de la aplicación y a la hora de procesar grandes cantidades de datos, por no mencionar que no podemos realizar la transformación de los datos.

Uno de los errores cometidos que puedo extraer de la gestión del proyecto, y aun siendo consciente de esto en un principio no he podido evitar, es la idea de empezar abarcando demasiado y pretender hacer muchas cosas, que posteriormente se han ido descartando o afinando por cuestiones de tiempo.

De esas características desechadas, algunas pueden implementarse en un futuro para mejorar este proyecto; entre ellas destaca la inclusión del gráfico de dendograma, la integración con Google Drive y sus APIs (pudiendo crear un dashboard exactamente igual que un documento de Google), aumentar el número de parámetros en la visualización de gráficos, o permitir aplicar transformaciones en los datos.

En definitiva, este TFG, me ha permitido ampliar mis conocimientos, y ser mucho más consciente de la inversión de tiempo y personal que supone la finalización de muchos proyectos software, incluso aquellos aparentemente “Sencillos” cuyo esfuerzo reside precisamente en esta apariencia de simplicidad.

Referencias

[1] MuyCanal. (2019). Las pymes temen la complejidad y costes de la analítica de datos. MuyCanal. Available: https://www.muycanal.com/2019/08/05/pymes-analitica-de-datos.

[2] Herrera, J. (2019). El mercado de Big Data y Business Analytics de aquí a 2022. Imf-formacion. Available: https://blogs.imf-formacion.com/blog/tecnologia/mercado-big-data-business-analytics-2022-201902.

[3] Negash, S., & Gray, P. (2008). Business intelligence. *In Handbook on decision support systems 2* (pp. 175-193). Springer, Berlin, Heidelberg. Available: https://link.springer.com/chapter/10.1007/978-3-540-48716-6_9.

[4] Sangüesa, J. (2019). El estado actual del Business Intelligence alojado en la nube. Allcloud. Available: https://allcloud.es/business-intelligence-nube-2019.

[5] Barbero, J. L., & Sánchez, L. (2006). PYMES en España. Fundación EOI. Available: http://www.eoi.es/nw/multimedia/PublicacionesEOI/2007_Libro_23.pdf.

[6] Medina-Chicaiza, R. P., Chiliquinga-Vejar, L. D. C., & Ortiz-Barba, A. P. (2016). Aproximación sobre la inteligencia de negocios en las PYME. *Dominio de las Ciencias, 2*(4), 370-382. Available: https://dominiodelasciencias.com/ojs/index.php/es/article/view/260.

[7] Moreno, H. (2017). Data Analytics Is No Longer A Nice Option -- It's The Core Of The Enterprise. Forbes. Available: https://www.forbes.com/sites/forbesinsights/2017/06/12/data-analytics-is-no-longer-a-nice-option-its-the-core-of-the-enterprise/#55b9101d77ec.

[8] Tableau. (2017). Tableau presenta nuevos precios por suscripción. Tableau, Seattle. Available: https://www.tableau.com/es-es/about/press-releases/2017/tableau-introduces-new-subscription-pricing.

[9] Losilla, J. (2013). Qlikview vs Tableau: Comparativa de herramientas de Business Intelligence. Porlaempresa. Available: https://porlaempresa.com/qlickview-vs-tableau-comparativa-de-herramientas-de-business-intelligence.

[10] Lavi, S. (2020). Pentaho Data Integration Pricing Guide (Jan 2020). Itqlick. Available: https://www.itqlick.com/pentaho-data-integration/pricing.

[11] Keller, J. (2019). Pentaho vs Tableau: Which Business Intelligence Software is the Winner?. Selecthub. Available: https://www.selecthub.com/business-intelligence/pentaho-vs-tableau.

[12] Datafalir. (2019). QlikView Pricing and Licensing – 3 Additional Services Offered By QlikView. Datafalir. Available: https://data-flair.training/blogs/qlikview-pricing-and-licensing.

[13] Microsoft. (2020). Precios de Power BI. Microsoft. Available: https://powerbi.microsoft.com/es-es/pricing.

[14] Virgo, J. (2019). Power BI vs QlikView – Business Intelligence Tool Comparison. Aptude. Available: https://aptude.com/blog/entry/power-bi-vs-qlikview-bi-tool-comparision.

[15] Golfarelli, M., Pirini, T., & Rizzi, S. (2017). Goal-based selection of visual representations for big data analytics. In *International Conference on Conceptual Modeling* (pp. 47-57). Springer, Cham. Available: https://link.springer.com/chapter/10.1007/978-3-319-70625-2_5.

[16] Luján, S., & Aragonés, J. (2012). Nuevos estándares en el desarrollo de sitios web: HTML5 y CSS3. Dlsi. Available: http://desarrolloweb.dlsi.ua.es/cursos/2012/nuevos-estandares-desarrollo-sitios-web/desarrollo-web-actual.

[17] Stackshare. (s. f.). amCharts vs AnyChart vs Highcharts. Stackshare. Available: https://stackshare.io/stackups/amcharts-vs-anychart-vs-highcharts.

[18] w3schools. (s. f.). HTML data-* Attributes. W3school. Available: https://www.w3schools.com/tags/att_global_data.asp.

[19] Roberto, C. (2019). Día mundial de la pyme, ¿qué representan para la economía de nuestro país?. Pymesyautonomos. Available: https://link.springer.com/chapter/10.1007/978-3-319-70625-2_5.

[20] IBM InfoSphere Information Server on Cloud: Pricing. (2020). IBM. Available: https://www.ibm.com/cloud/information-server/pricing.